510.84
Il6r
no. 587
Cop 2

UIUCDCS-R-73-587

# GUIDE-O
# -- AN EXPERIMENTAL INFORMATION SYSTEM --

by

Shinnichi Murai

August, 1973

DEPARTMENT OF COMPUTER SCIENCE
UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN · URBANA, ILLINOIS

# GUIDE-O
# -- AN EXPERIMENTAL INFORMATION SYSTEM --

BY

SHINNICHI MURAI

B. Eng., Kyoto University, 1963

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Computer Science
in the Graduate College of the
University of Illinois at Urbana-Champaign, 1973

Urbana, Illinois

## ACKNOWLEDGMENT

The author wishes to thank Professor Jurg Nievergelt for providing me with the opportunity to work on the project and for many valuable suggestions and encouragements on the work. Thanks are also due to Mr. Dave Eland for suggestions and corrections on the manuscript and Messrs. Jean Pradels and Mateti Prabhaker for discussions on the project. Special thanks go to Mrs. Connie Slovak and June Wingler for the quick and excellent typing and preparation of the paper.

## TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

# I. INTRODUCTION

The project to automate a portion of the introductory computer science courses onto the PLATO IV system [2] has been initiated in order to meet the need to offer those courses to a sharply growing number of people whose background and ability differ considerably [1]. The automation of instruction has two aspects, quantitative and qualitative. The automation of grading exams can be considered as an example of quantitative effect. Many lessons in PLATO IV can be said to have qualitative effect in the sense that they allow personal and interactive communication with each student. The main objective of GUIDE [1, 3], a conversational information system, is to help students in selecting PLATO IV lessons to study. This selection is made in an interactive manner, and is based on the student's past activity and performance. The necessity of this kind of system is apparent, considering the rapid growth of the number of students with different background and ability, and also the increasing number of lessons in the PLATO IV system. GUIDE-0 is intended to serve as an experimental system for GUIDE to investigate the formation of search prescriptions, keyword repertoire, data items to be included in the data base, structure of the data base, search algorithms, etc. in the PLATO IV environment. It is also intended to fulfill the practical necessity for GUIDE until it is actually implemented. The capabilities of GUIDE-0 are "reduced" in the following sense:

(a) The language for the communication between GUIDE-0 and its users is severely restricted. GUIDE will allow communication by natural language (English), but GUIDE-0 is restricted to its own special language (instructions).

(b) No judging function based on each student's past record is included in GUIDE-0. It displays only the information in the data base. For example, students can't ask GUIDE-0 such questions as "What should I study today?". A student is expected to ask GUIDE-0 to display his past record and/or course outline, and to judge <u>by himself</u> what to study.

(c) GUIDE-0 is not "clever" enough. It searches the data base based upon exactly the user specified terms. For example, suppose a user asked to search the lessons which deal with "2-dimensional array<u>s</u>", and also suppose that not "2-dimensional array<u>s</u>" but "2-dimensional array" is included in the keyword repertoire. Then, GUIDE-0 can't search by the plural form. Also, GUIDE-0 won't automatically do the expansion of a search prescription to a broader term, for example, from "2-dimensional array" to "array", although GUIDE-0 suggests users to do so, when it can't find what users asked.

(d) The scope of the data base is small, and the structure of the data base is influenced by the scope and current status of the ever expanding PLATO IV system architecture.

# II. THE FUNCTIONS OF GUIDE-0

## 1. Search for and Display Lessons Which Match a Given Search Prescription

Users are supposed to give GUIDE-0 a search prescription, i.e., a logical expression of keywords which represents their interest. Then GUIDE-0 searches the data base for the lessons which match the given search prescription, and displays the names and abstracts of those lessons.

### 1.1 Input (Search Prescription)

The input for this function is a logical expression of keywords followed by a semicolon. The allowed logic operations in the expressions are AND(*), OR(+) and NOT('). Operators and operands may be separated by blanks, and parantheses (nested to any level) are allowed.

For example, suppose that a user wants to study about data structures other than arrays, in any languages other than PL/1 and ALGOL. Then the search prescription for him could be as follows:

```
data structure * array'*(pl/1 + algol)';
```

### 1.2 Output (Lesson Names and Abstracts)

The output of this function is a list of lesson names and abstracts which match the given search prescription. The following is an example:

| LESSON | ABSTRACT |
|---|---|
| racetrack | Simulation Experiment |
| somaga | Software Management Game to teach Programming |
| montecarlo | Area Calculation by Monte Carlo Method |

If the number of lessons to be displayed exceeds the display size, the lessons are paged and the first page is displayed at first. "NEXT" and "BACK" keys are used to turn pages forward or backward.

### 1.3 Notes

Each keyword in the search prescription must be exactly the one listed in the Keyword Table of GUIDE-0. As explained in the introduction, for example, if only the singular form of a keyword is listed and if a user specifies the plural form, GUIDE-0 does not accept the search prescription and tells the user that the keyword is not included in the repertoire.

If GUIDE-0 cannot find any lessons which match the search prescription given, GUIDE-0 suggests that the user search by a broader term.

If there are syntactic errors in a search prescription, GUIDE-0 asks the user to correct it.

## 2. Display Lesson Descriptors

The user is asked to give a lesson name. Then GUIDE-0 searches the data base for the specified lesson, and displays its lesson descriptors. The lesson descriptors consist of the following items:

(a) Lesson name -- the name of the lesson which is registered in PLATO IV system.

(b) Type -- the type of the lesson such as "practice", "examination", etc.

(c) Abstract - a brief explanation about the contents of the lesson.

(d) Subject category -- each lesson included in the data base of GUIDE-0 is classified into one or more of the categories listed in Table 1.

0. GENERAL

1. INTRODUCTION
   - 1.0 General
   - 1.1 Programming
   - 1.2 Computers
   - 1.3 Applications

2. PROGRAMMING LANGUAGES
   - 2.0 General
   - 2.1 PL/I
   - 2.2 Fortran
   - 2.3 Basic

3. PROGRAMMING LANGUAGE CONCEPTS
   - 3.1 Introductory
     - 3.11 General Information
       - 3.111 Operating Systems -- Hasp, Express
       - 3.112 Compilers -- PL/I, PL/C
       - 3.113 Job Control Language
     - 3.12 Overview of a language
       - 3.121 General Program Format -- statements, labels, comments, line continuation, character sets, delimiters, separators, use of blanks, punctuation
       - 3.122 Program Structure -- sequencing rules, blocks, groups
       - 3.123 Basic Instruction Types -- executable unit
       - 3.124 Identifiers, Keywords, Reserved Words
       - 3.125 Interaction with Operating System and Environment
       - 3.126 Special Conventions, Features
       - 3.127 Illustrative Programs
   - 3.2 Data Types
     - 3.21 Arithmetic -- constants, variables, fixed, float, precision
     - 3.22 Character String
     - 3.23 Bit String
     - 3.24 Pointer
     - 3.25 Labels
     - 3.26 Declarations, Attributes
   - 3.3 Data Operations
     - 3.31 Arithmetic
       - 3.311 Operators
       - 3.312 Assignment Statement
       - 3.313 Expressions
       - 3.314 Initialization
       - 3.315 Built-in Functions
     - 3.32 String
       - 3.321 Operators
       - 3.322 Built-in Functions
     - 3.33 Comparison
       - 3.331 Operators
       - 3.332 Expressions
     - 3.34 Logical
       - 3.341 Operators
       - 3.342 Expressions
   - 3.4 Data Structures
     - 3.41 Arrays
     - 3.42 Hierarchical Structures
     - 3.43 List
   - 3.5 Data Storage
     - 3.51 Storage Types -- static, external, automatic, based, . . .
     - 3.52 Dynamic Storage Allocation
   - 3.6 Control Statements
     - 3.61 Unconditional Branch -- goto
     - 3.62 Conditional Branch -- computed goto, if, then, else
     - 3.63 Loops, Iteration -- do, while
     - 3.64 Recursion
   - 3.7 Input/Output
     - 3.71 Data formats -- (F, E, A, . . .)
     - 3.72 Printer carriage control -- page, line, column
     - 3.73 Types of I/O
       - 3.731 Stream
       - 3.732 Record
       - 3.733 Internal -- (get string, put string)
   - 3.8 Subprograms
     - 3.81 Functions
       - 3.811 Built-in -- arithmetic, string
       - 3.812 Library
       - 3.813 Programmer Defined -- internal, external
       - 3.814 Generic
       - 3.815 Recursive
     - 3.82 Subroutines
   - 3.9 Others
     - 3.91 Interrupts
     - 3.92 Debugging aids
     - 3.93 Compiler Directives, Macros

4. APPLICATIONS
   - 4.0 General
   - 4.1 Natural Sciences
   - 4.2 Engineering
   - 4.3 Social Sciences
   - 4.4 Humanities
   - 4.5 Business Data Processing
   - 4.6 Education

5. TECHNIQUES
   - 5.0 General
   - 5.1 Numerical Methods
     - 5.10 General
     - 5.11 Error Analysis; Computer Arithmetic
     - 5.12 Function Evaluation
     - 5.13 Interpolation; Functional Approximation
     - 5.14 Linear Algebra
     - 5.15 Nonlinear and Functional Equations
     - 5.16 Numerical Integration and Differentiation
     - 5.17 Differential Equations
     - 5.18 Integral Equations
   - 5.2 Combinatorial
     - 5.20 General
     - 5.21 Enumeration
     - 5.22 Sorting
     - 5.23 Searching
     - 5.24 Graphs
   - 5.3 Optimization
   - 5.4 Simulation
     - 5.40 General
     - 5.41 Discrete
     - 5.42 Continuous
     - 5.43 Monte Carlo
   - 5.5 Heuristic
     - 5.50 General
     - 5.51 Heuristic Search
     - 5.52 Adaptive Programs
     - 5.53 Pattern Recognition
   - 5.6 Programming Techniques
     - 5.61 List Processing

Table 1. Subject Category

(e) Keywords -- keywords which represent the contents of the lesson.

(f) Time required -- the estimated time required to go through the lesson.

(g) Relations to other lessons -- the relations between lessons such as "prerequisite of the lesson", "sequel to the lesson", etc.

3. Display Course Outline

The user specifies the course and section number. Then GUIDE-0 displays the course outline of the specified course. The Course Outline is a list of lesson names with the dates by which each lesson must be taken, the estimated time required to go through the lesson, and the expected performance in the lesson (if it is a "practice", "exercise" or "exam" type lesson).

4. Display Student Record

The user specifies the course and section number, his name and his social security number. Then GUIDE-0 displays his record in the specified course. The student record is a list of lesson names with the last date the student took each lesson, the time the student spent on the lesson, and his performances in the lesson (if it is a "practice", "exercise" or "exam" type lesson).

# III. THE DATA BASE OF GUIDE-O

The data base for GUIDE-O consists of two parts, the Lesson Catalog and Course Record, each of which consists of several files. The Lesson Catalog contains the information related to lessons such as the abstracts of the lessons, the keywords attached to the lessons, etc. and is mainly used for functions 1. and 2. of the preceding chapter. The Course Record contains the information related to course activity such as course outlines, each student's performance in each lesson, etc. and is used for functions 3. and 4. The Lesson Catalog consists of three files: Lesson Catalog 1, Lesson Catalog 2 and Keyword Table. The Course Record consists of four files: Course Directory, Course Outline, Student Directory and Student Records. All of these files are stored in the "common" storage provided by PLATO IV [4]. Figures 3.1 and 3.2 illustrate the structure of the Lesson Catalog and Course Record.

## 1. Lesson Catalog

The Lesson Catalog provides the information necessary to know what a lesson is about, or to retrieve the lessons which are supposed to be related to a user's interest. The Lesson Catalog consists of three files: Lesson Catalog 1, Lesson Catalog 2 and Keyword Table. The first two are related to what a lesson is about, and the last is used for retrieval purposes. The difference between Lesson Catalog 1 and Lesson Catalog 2 is as follows:

(a) Lesson Catalog 2 is sorted by lesson name, thus allowing a binary search by lesson name; on the

Lesson Catalog 1

| Lesson Name | Abstract | Subject Category | Keywords | Not Used |
|---|---|---|---|---|
| montecarlo | Area Calculation by Monte-Carlo Method | 5230 1300 | $\underbrace{K_1}_{4b} K_2 \ldots K_7$ | |
| 10 chars | 62 characters | 2x4 chars | 7x8 bits | 4 bits |
| 1 word | 7 words | | 1 word | |
| 9 words | | | | |

Lesson Catalog 2

| Lesson Name | No. | Time Required | Relations to Other Lessons | Not Used | Type |
|---|---|---|---|---|---|
| files | 14 | $t_1$ | $\underbrace{\rho_1}_{4b}\underbrace{l_1}_{6b} \rho_1 l_2 \rho_2 l_3 \ldots$ | | $y_1$ |
| montecarlo | 1 | $t_2$ | | | $y_2$ |
| pl1 data | 17 | $t_3$ | | | $y_3$ |
| 10 chars | 6 bits | 6 bits | 4x(4+6) bits | 4 bits | 4 bits |
| 1 word | 1 word | | | | |
| 2 words | | | | | |

Keyword Table

| Keyword | Retrieval Code |
|---|---|
| array | 1010......011 |
| assignment statement | 0100.....1000 |
| 20 characters | 60 bits |
| 2 words | 1 word |
| 3 words | |

Figure 3.1 File Structure of Lesson Catalog

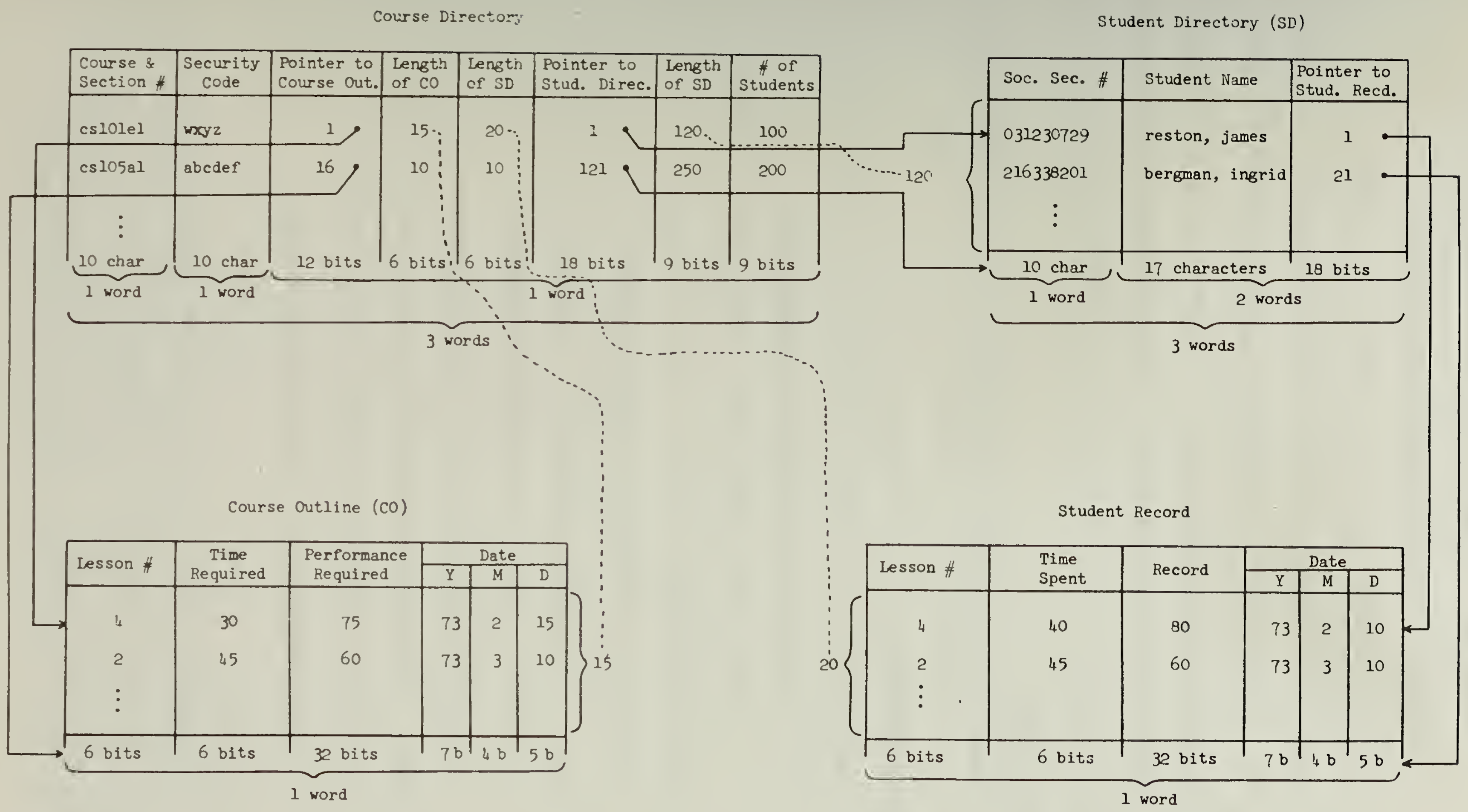

Figure 3.2 File Structure of Course Record

other hand, Lesson Catalog 1 is sorted by lesson number. (Actually no sorting operation is done on Lesson Catalog 1. The location of a lesson in Lesson Catalog 1 is the lesson number of the lesson.)

(b) Lesson Catalog 2 contains different information about the same lessons which are stored in Lesson Catalog 1.*

At the current implementation, the information for up to 60 lessons can be stored in the Lesson Catalogs and up to 256 keywords can be stored in the Keyword Table.

Figure 3.1 illustrates the structure of the Lesson Catalog.

## 1.1 Lesson Catalog 1

Lesson Catalog 1 consists of the following four fields:

(a) Lesson Name (lessonml) -- 10 characters

This field contains the lesson name of maximum 10 characters.

(b) Abstract (abstrct) -- 62 characters

This field contains the very brief explanation of the lesson of maximum 62 characters.

(c) Subject Category -- 2x4 characters

This field contains up to two codes each of which consists of 4 digits (characters) and represents the category of the lesson.

*Originally, Lesson Catalog 2 was intended to provide only an index to Lesson Catalog 1. However, since the unit of information processing in the PLATO IV system (in the TUTOR language) is basically a word (not a character or a byte) and the available storage area is severely restricted, it was decided to store a part of the information about lessons into the area of Lesson Catalog 2 which would otherwise be wasted [4, 5].

(d) Keyword -- 7x8 bits

This field contains up to seven keyword identity codes (the location of the keyword in Keyword Table) of 8 bits. Thus a maximum of 8 keywords can be attached to each lesson.

Thus Lesson Catalog 1 consumes 9 words of memory per lesson, i.e., 640 words are necessary for 60 lessons.

## 1.2 Lesson Catalog 2

Lesson Catalog 2 consists of the following five fields:

(a) Lesson Name (lessnm2) -- 10 characters

This field contains the lesson name of up to 10 characters.

(b) Lesson Number (glessnn) -- 6 bits

This field contains the location of the lesson in Lesson Catalog 1 (the lesson number).

(c) Time Required (gtime) -- 6 bits

This field contains the average time required to finish the lesson.

(d) Relations to Other Lessons -- 4x(4+6) bits

This field contains up to 4 relation identity codes of 4 bits each of which is followed by the lesson number (6 bits) of the lesson which has the relationship specified by the relation identity code with the lesson specified in the Lesson Name field. For example, if the lesson has two prerequisites 12 and 13, and one sequel 14; the Relation field should look like

$$\underbrace{\rho_1}_{4\text{ bits}}\;\underbrace{12}_{6\text{ bits}}\;\underbrace{\rho_1}_{\cdots}\;\underbrace{13}_{\cdots}\;\underbrace{\rho_2}_{\cdots}\;\underbrace{14}_{\cdots}\;\;\underbrace{000\ldots0}_{10\text{ bits}}$$

where $\rho_1$, $\rho_2$ are the relation identity codes for "prerequisite" and "sequel" respectively, and 11, 12, 13 are lesson numbers.

(e) Type (gtype) -- 4 bits

This field contains the lesson type identity code of 4 bits.

Lesson Catalog 2 consumes 2 words per lesson, thus 120 words per 60 lessons.

## 1.3 Keyword Table

The Keyword Table consists of the following two fields:

(a) Keyword (keyword) -- 20 characters

This field contains a keyword (or phrase) of up to 20 characters.

(b) Retrieval Code (retcode) -- 60 bits

This field contains a retrieval code of 60 bits which is attached to the keyword specified in the Keyword field. Each bit of a retrieval code represents a lesson and the bit position corresponds to the lesson number. For example, consider the following case:

| Keyword | Retrieval Code |
|---|---|
| program flow control | 00100001100.....0 |

This indicates that the keyword "program flow control" is attached to three lessons, the lesson numbers of which are 3, 8 and 9 respectively. Thus if the user specifies the keywords which represent his interest, then the GUIDE-0 program searches the Keyword Table, obtains the retrieval codes, and gives the necessary information (lesson names and abstracts) about the lessons which appear in the retrieval codes.

The Keyword Table is sorted alphabetically by keywords, allowing a binary search for a given keyword. The Keyword Table consumes 3 words of memory per keyword, thus 768 words per 256 keywords.

## 2. Course Record

The Course Record provides students and instructors the information such as course outlines, students' performances in some lessons, the date when students took lessons, etc. The Course Record consists of the following four files:

(a) Course Directory,

(b) Course Outline

(c) Student Directory,

(d) Student Record.

The Course Directory contains the pointers to course outlines and to student directories, and some other administrative information. The Course Outline contains course outlines (the schedule of lessons to be taken in each course). The Student Directory contains lists of students enrolled in the courses and pointers to each student's own record. The Student Record contains students' records such as performance in lessons, time spent in lessons, etc.

### 2.1 Course Outline

The Course Outline is the schedule of lessons which are to be taken by students who are enrolled in the course. It consists of the following five fields:

(a) Lesson Number -- 6 bits

This field contains a lesson number (the location of the lesson in Lesson Catalog 1).

(b) Time Required -- 6 bits

This field contains the average time or the maximum time to finish the lesson specified in the Lesson Number field.

(c) Performance Expected -- 32 bits

This field contains the performance expected of the lesson specified in the Lesson Number field.

(d) Date -- 16 bits

This field contains the date by which students are expected to finish the lesson specified in the Lesson Number field. The first 7 bits contain the year, the next 4 bits the month, and the last 5 bits the day.

Thus, the Course Outline occupies 2 words of memory per lesson.

## 2.2 Student Record

The Student Record stores the various student records such as the performance in a lesson, the last date the student took the lesson, etc. for all students who are enrolled in the courses listed in Course Directory. The Student Record has exactly the same format as the Course Outline, consisting of the following 4 fields:

(a) Lesson Number -- 6 bits

This field contains a lesson number (the location of the lesson in Lesson Catalog 1).

(b) Time Spent -- 6 bits

This field contains the time spent by a student to finish the lesson specified in the Lesson Number field.

(c) Records -- 32 bits

This field contains coded records of the student's performance in the lesson specified in the Lesson Number field.

(d) Date -- 16 bits

This field contains the last date the student took the lesson specified in the Lesson Number field.

The Student Records occupy 2 words of memory per lesson (same as Course Outline).

## 2.3 Student Directory

The Student Directory contains lists of students who are enrolled in the courses listed in the Course Directory, and the pointers to each student's student record. Each course has its own student directory and is sorted by the social security number of the students who are enrolled in the course. If the same student takes two different courses, his name appears twice in two student directories. The Student Directory consists of the following three fields:

(a) Social Security Number -- 9+1 digits (characters)

This field contains the social security number of a student as a character string of 9 digits (characters) +1 blank character.

(b) Student Name -- 17 characters

This field contains a student's name of up to 17 characters.

(c) Pointer to Student Record -- 18 bits

This field contains a pointer to the student record of the student specified in Social Security Number and

Student Name field. The logical location or array subscript, not the physical address, is meant by "pointer".

Thus, the Student Directory occupies 3 words of memory per student.

## 2.4 Course Directory

The Course Directory contains various administrative data, consisting of the following eight fields:

(a) Course and Section Number (coursen) -- 10 characters

This field contains course and section number (e.g. cs101e1, math105a1, etc.).

(b) Security Code (seccode) -- 10 characters

This field contains security code of up to 10 characters, which protects the privacy of student records.

(c) Pointer to Course Outline (pcoutln) -- 12 bits

This field contains a pointer to the course outline of the course specified by Course and Section Number field. The logical location or array subscript, not the physical address, is meant by "pointer".

(d) Length of Course Outline (lcoutln) -- 6 bits

This field contains the length of the course outline, i.e., the number of lessons contained in the course outline for the course specified in Course and Section Number field.

(e) Length of Student Records (lsrecrd) -- 6 bits

This field contains the length of student records, i.e., the maximum number of lessons to be recorded in each student record. The number contained in this field should be equal to or greater than that of Length of Course Outline. If both numbers are equal, only the lessons listed in the Course Outline are recorded in the Student Record.

(f) Pointer to Student Directory -- 18 bits

This field contains a pointer to the student directory of the course specified in Course and Section Number field.

(g) Length of the Student Directory -- 9 bits

This field contains the length of the student directory of the course specified in Course and Section Number field, i.e., the <u>maximum</u> number of students to be enrolled in the course.

Note that this number is used to reserve a space for the student directory at the beginning of a semester. No more students than this number can be registered in the course under the current version of the GUIDE-0 editor.

(h) Number of Students -- 9 bits

This field contains the current number of students who are registered in the course specified in Course and Section Number field.

Thus, the Course Directory occupies 3 words of memory per course.

## 3. Implementation of the Data Base

### 3.1 PLATO IV Storage Organization [4, 5]

The Storage hierarchy of the PLATO IV system consists of three levels (see Figure 3.3)

(a) Central Memory (CM),

(b) Extended Core Storage (ECS),

(c) Disk.

Central Memory is used for the lesson which is currently being executed by the CPU, and so the contents of central memory stays the same only within a single time slice. Extended Core Storage is used to store the lessons which are being "taken" by students currently sitting at a terminal. For example, if 20 students are taking 5 different lessons, those 5 lessons are stored in ECS at the same time. The Disk is used as a permanent storage device for all lessons in the PLATO IV system.

Each lesson in PLATO IV has access to two kinds of variables, "student variables" and "common variables". A set of 150 student variables is attached to each student and is stored in disk. A set of maximum 4186 common variables is attached to a lesson (if necessary) and is also stored in disk. Whenever a lesson is condensed, the student variables attached to the student who is going to use the lesson, and the common variables attached to the lesson being condensed are transferred into ECS from disk. When a time slice is given to a student by the system program of PLATO IV, the lesson the student is working on, the student variables of the student, and in case of automatic loading mode the first 1500 out of 4186 common variables attached to the lesson (if any) are loaded into the central memory. In case of non-automatic loading mode, the loading of up to 1500 common variables is specified by the instruction in a lesson. In this case you can specify

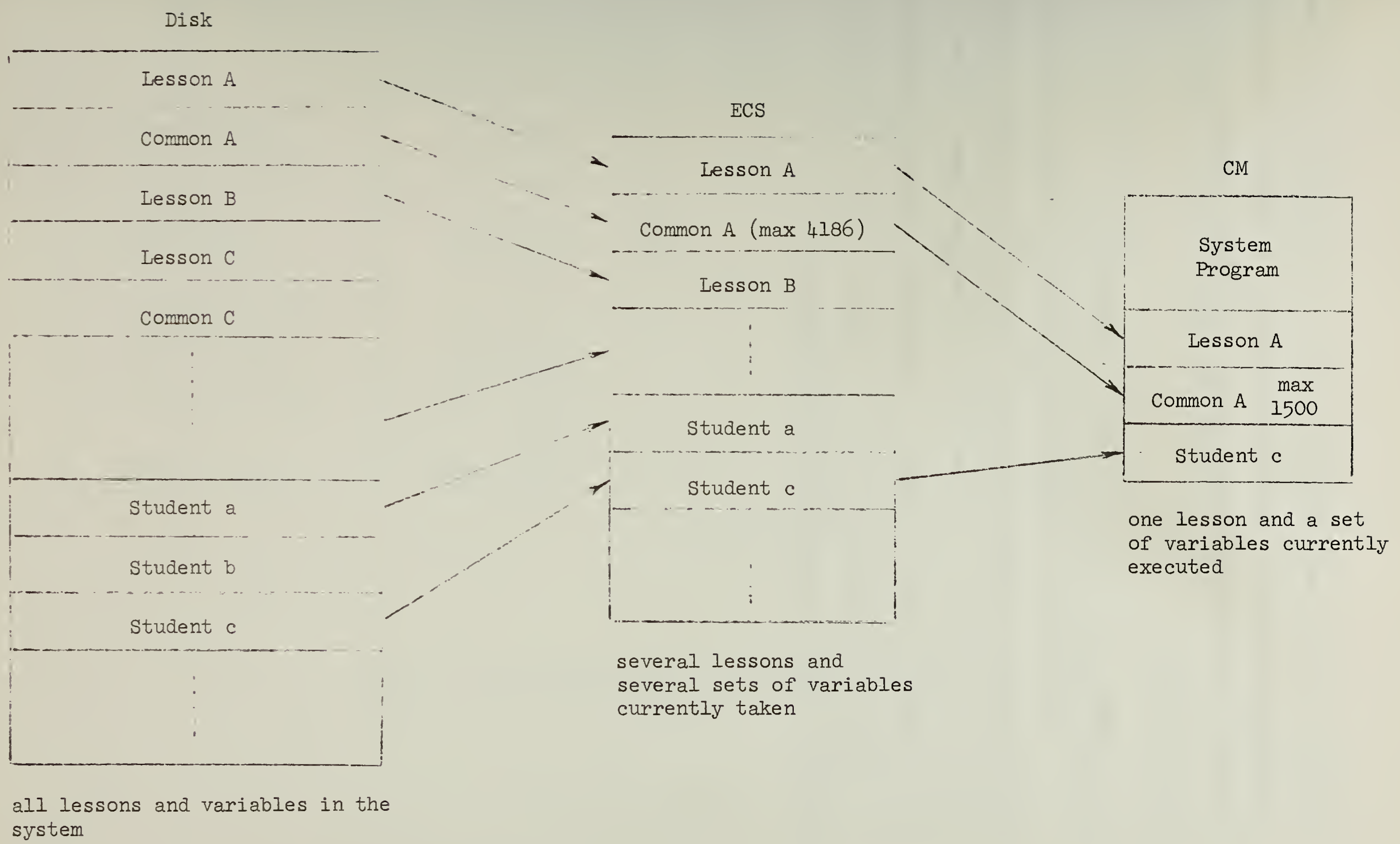

Figure 3.3 Storage Hierarchy of PLATO IV

which part of 4186 common variables to load. In other words, even though 150 student variables and eventually 4186 common variables can be accessed by a lesson, only 150 student variables and 1500 common variables can be accessed at the same time.

## 3.2 Implementation of the Data Base in Common Storage

The data base of GUIDE-0 is implemented in common storage area. Since the amount of storage area which is available as "common" storage is limited to 4186 words as explained above, the maximum number of items of each file is limited as shown in Table 2. This would be acceptable considering the experimental nature of GUIDE-0. The amount of storage used and the location of each file are also shown in Table 2.

| File | Max. No. of Items | Amount of Storage | Location |
|---|---|---|---|
| Lesson Catalog 1 | 60 lessons | 540 (words) | 1501 ~ 2040 |
| Lesson Catalog 2 | 60 lessons | 120 | 2041 ~ 2160 |
| Keyword Table | 256 keywords | 768 | 2161 ~ 2928 |
| Course Directory | 10 courses | 30 | 1 ~ 30 |
| Course Outline | 150 lessons | 150 | 3001 ~ 3150 |
| Student Directory | 150 students | 450 | 31 ~ 480 |
| Student Record | 1020 lessons | 1020 | 481 ~ 1500 |

Total = 3078 words

Table 2. The Implementation of GUIDE-O Data Base

# IV. THE MODULES OF GUIDE-0

GUIDE-0 consists of the following 8 major modules:

1. Instruction Decoder and Controllers (idecode)
2. Lexical Analyzer of Search Prescription (lexi)
3. Syntax and Semantics Analyzer of Search Prescription (parser)
4. Search Range Vector Calculator (calcsrv)
5. Sequential Search Module (ssearch)
6. Binary Search Module (bsearch)
7. Message Editors
8. Miscellaneous Modules

The modules other than 1 are called (joined) as subroutines by module 1 or the others, and the controllers (parts of 1) are basically sequences of subroutine-calls. The main flow of GUIDE-0 and the structural relationship between modules are shown in the following section.

## 1. Instruction Decoder and Controllers

This module is further subdivided into 5 submodules:

(a) Instruction Decoder

(b) Controller 1 (search for lessons which match search prescription)

(c) Controller 2 (display lesson descriptors)

(d) Controller 3 (display course outline)

(e) Controller 4 (display student record)

The main flow of GUIDE-0 can be shown in terms of these 5 sub-modules as in Figure 4.1.

## 1.1 Instruction Decoder "idecode"

The Instruction Decoder displays the four functions of GUIDE-0 described in Chapter 3 to the user and asks him to choose one of them. According to his response, the Instruction Decoder jumps to one of the following four controllers.

## 1.2 Controller 1 (Search for Lessons Which Match Search Prescription) "idclsp"

As shown in Figure 4.2, Controller 1 receives a search prescription as described in II.1.1. The search prescription is stored in the array "sprescr". Then Controller 1 joins (calls) the Syntax and Semantics Analyzer. The Syntax and Semantics Analyzer parses the search prescription stored in "sprescr" and puts the result into the array "postfix". The result is a postfix notation with keywords as operands and three kinds of logic operators (AND, OR, NOT) as operators, as seen by the name of the array.

Next, Controller 1 joins the Search Range Vector Calculator. The Search Range Vector Calculator goes through "postfix" and calculates (via the logical operations AND, OR, NOT) the search range vector and puts the resultant vector into a location of the array "tsrange". The location is specified by the first location of the array "pstack", i.e., pstack(1).

After this, Controller 1 joins the Message Editor 1. Message Editor 1 interprets the final search range vector and generates the display image.

Figure 4.3 illustrates the flow of control among the modules used for the function 1.

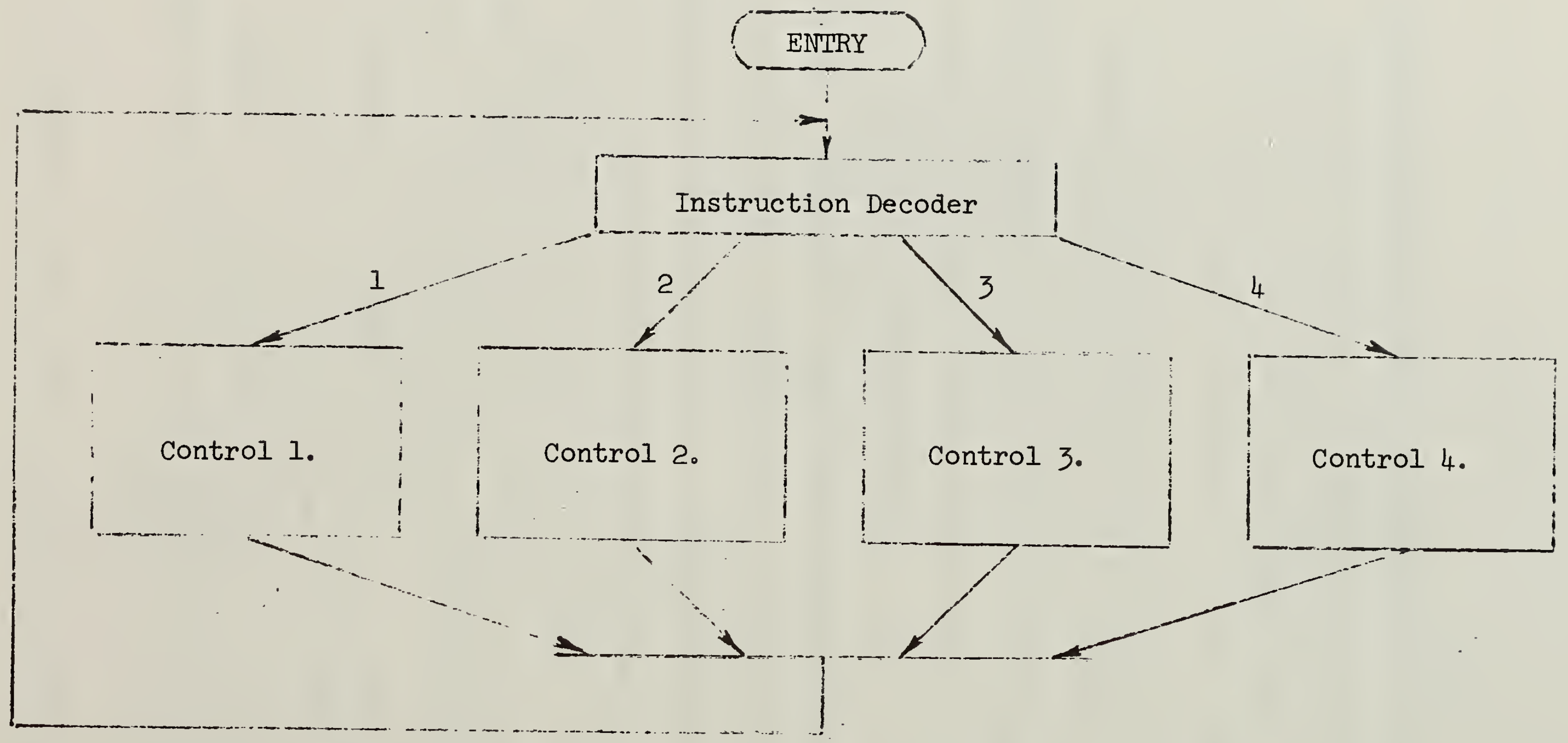

Figure 4.1 Main Flow of GUIDE-0

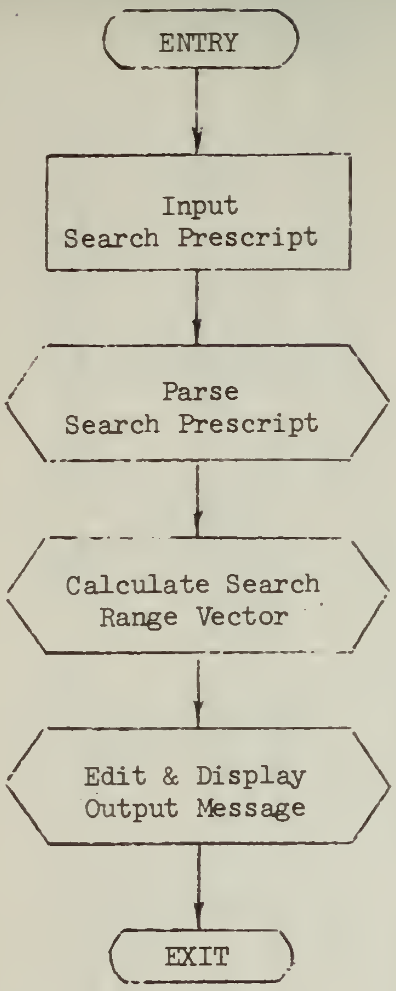

Figure 4.2 Flow of Controller for Function 1.

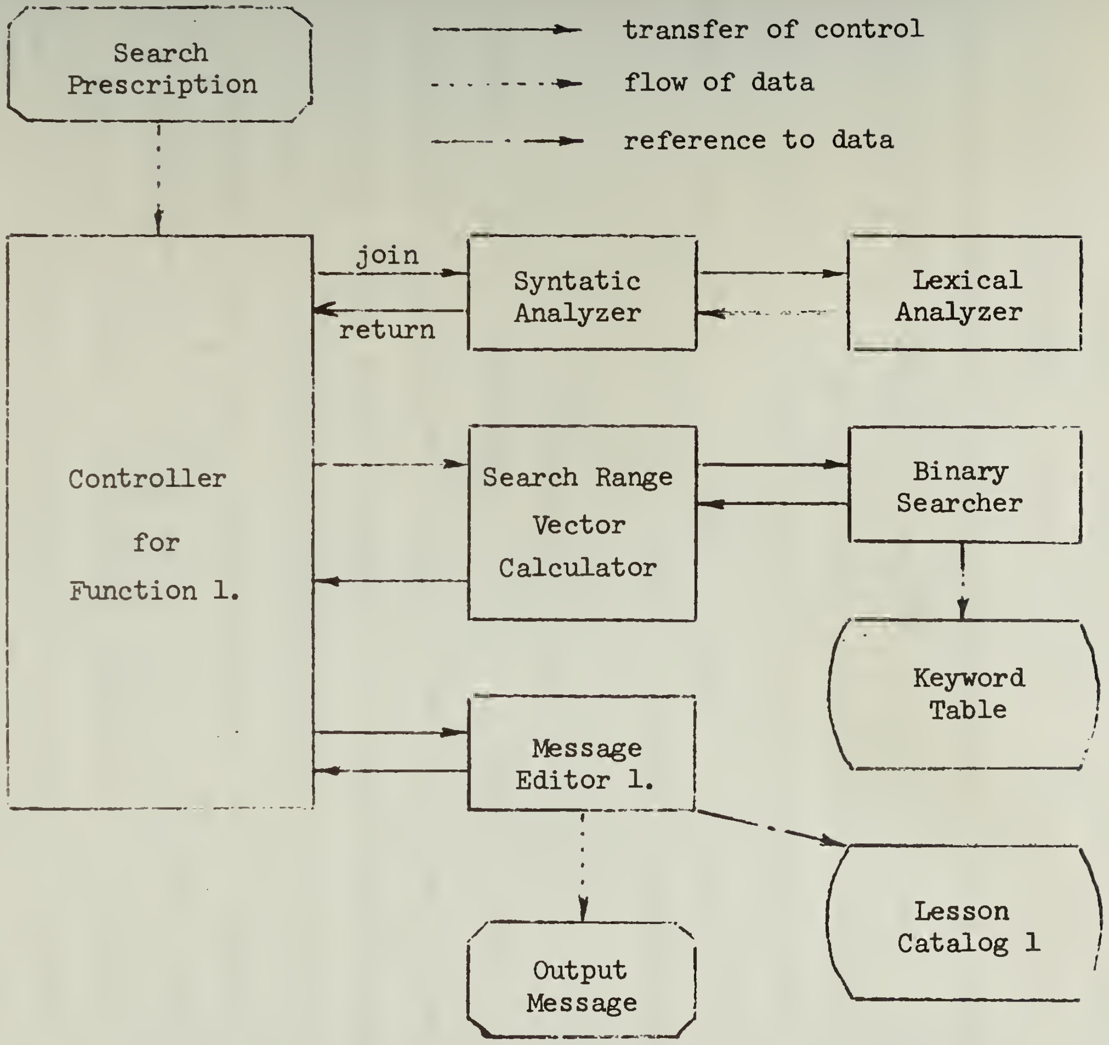

Figure 4.3 Structural Relationship between Modules - 1.

## 1.3 Controller 2 (Display Lesson Descriptors) "idcdld"

As shown in Figure 4.4 and Figure 4.5, Controller 2 receives a lesson name, and joins the Binary Search module. The Binary Search module searches the Lesson Catalog 2 for the specified lesson, and returns the "logical" location of the lesson in Lesson Catalog 2.

Then Controller 2 joins the Message Editor for Function 2. The message editor reads the necessary data about the lesson in Lesson Catalogs 1 and 2, and generates the display image.

## 1.4 Controller 3 (Display Course Outline) "idcdco"

As shown in Figure 4.6 and 4.7, Controller 3 receives a course and section number whose course outline the user wants to know, and then joins the Sequential Search module. The Sequential Search module searches the Course Directory for the specified course and section, and returns the "logical" location of the course and section in the Course Directory.

Next, Controller 3 joins Message Editor 3. Message Editor 3 generates that part of the display image which is unique to the course outline, obtains the location of the course outline from the Course Directory, and then joins the Subeditor which is shared with Message Editor 4.* The Subeditor reads the specified location of the Course Outline and Lesson Catalog 1, and generates the rest of the display image.

## 1.5 Controller 4 (Display Student Record) "idcdsr"

As shown in Figure 4.8 and 4.9, Controller 4 receives the course and section number in which the student is enrolled, and joins the Sequential Search module. The Sequential Search module searches the Course Directory

---

*The Course Outline and Student Record have the same file structure.

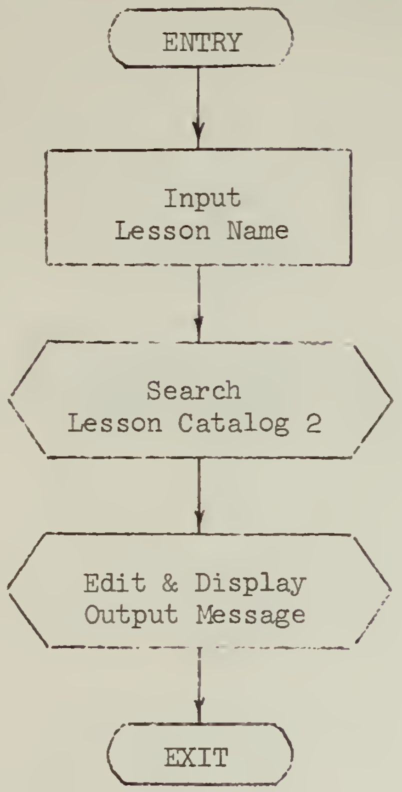

Figure 4.4 Flow of Controller 2

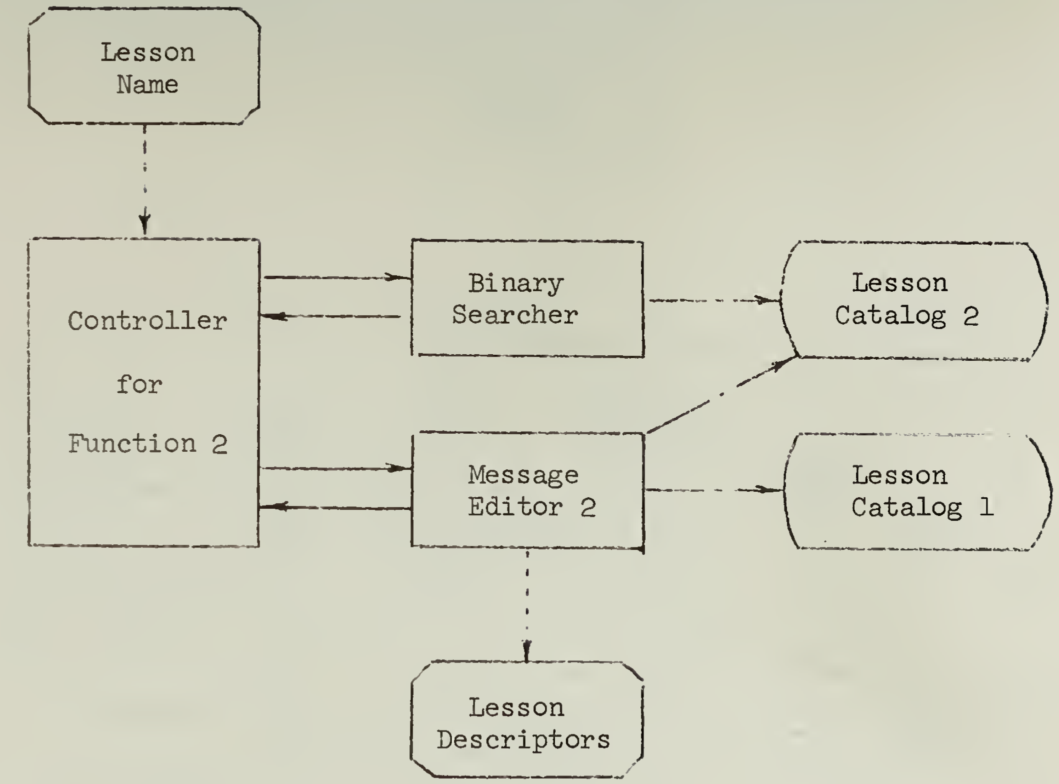

Figure 4.5 Structural Relationship between Modules - 2

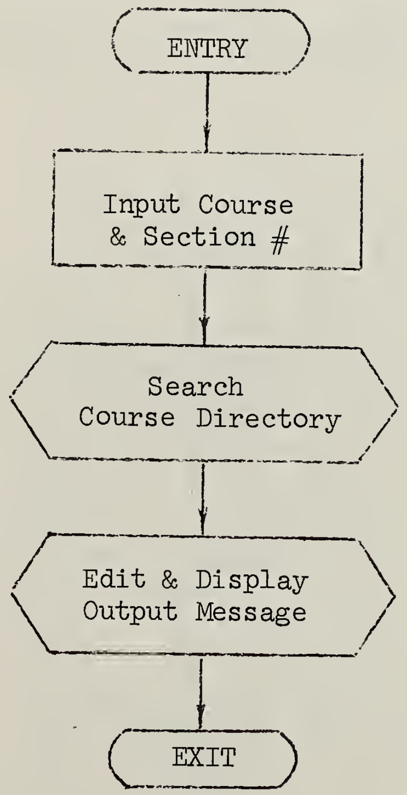

Figure 4.6 Flow of Controller 3

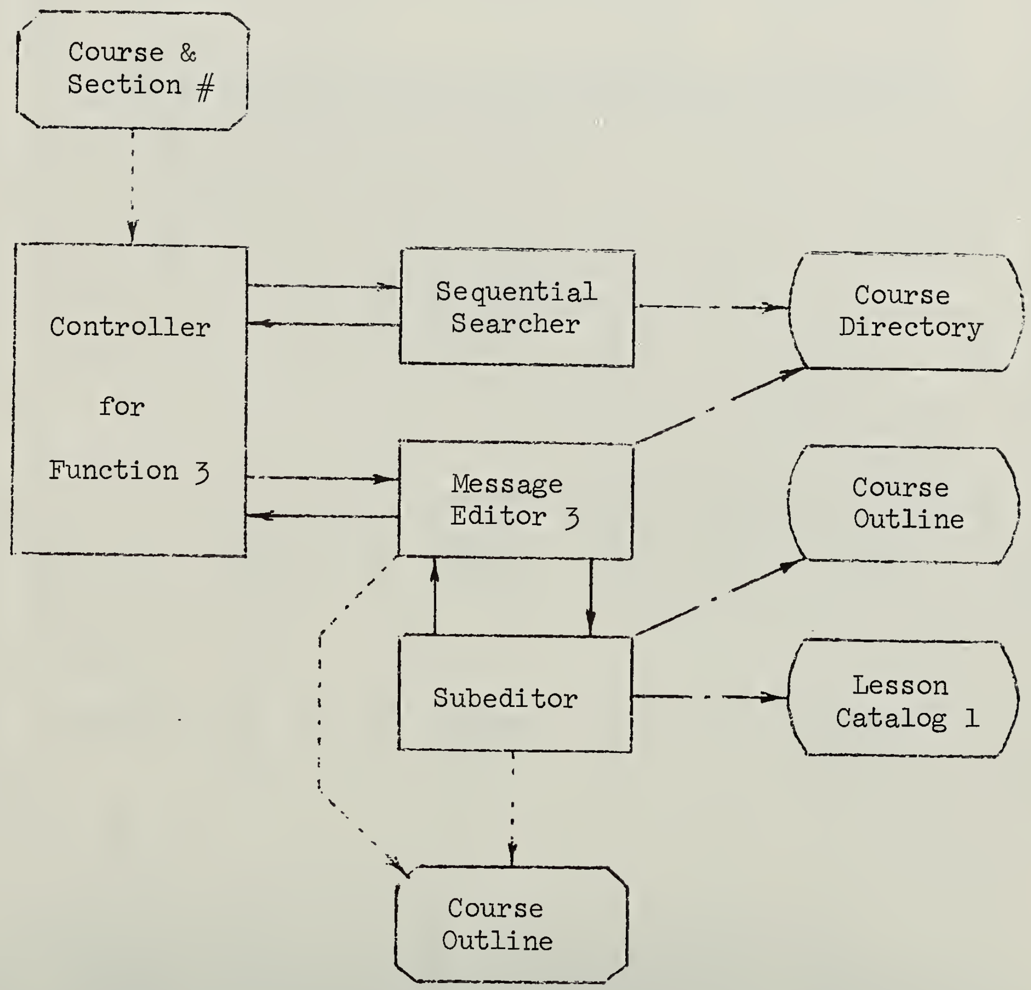

Figure 4.7 Structural Relationship between Modules - 3

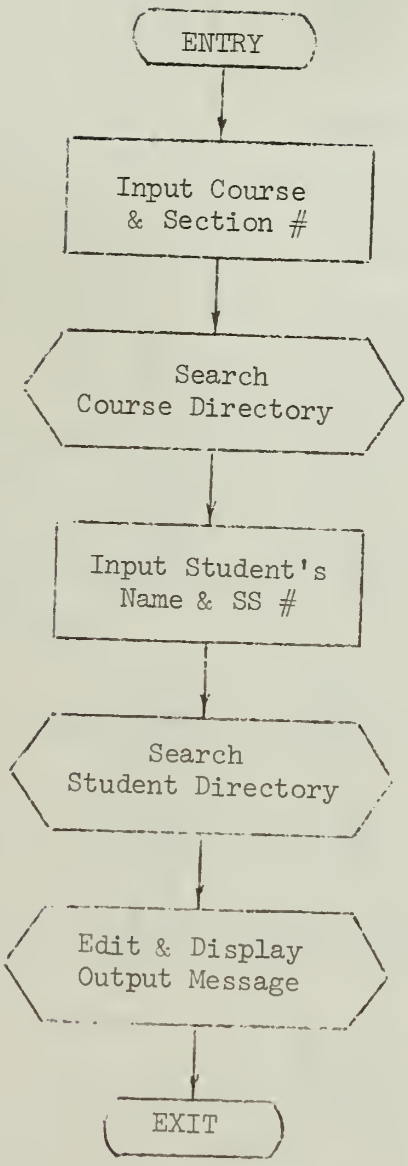

Figure 4.8 Flow of Controller 4

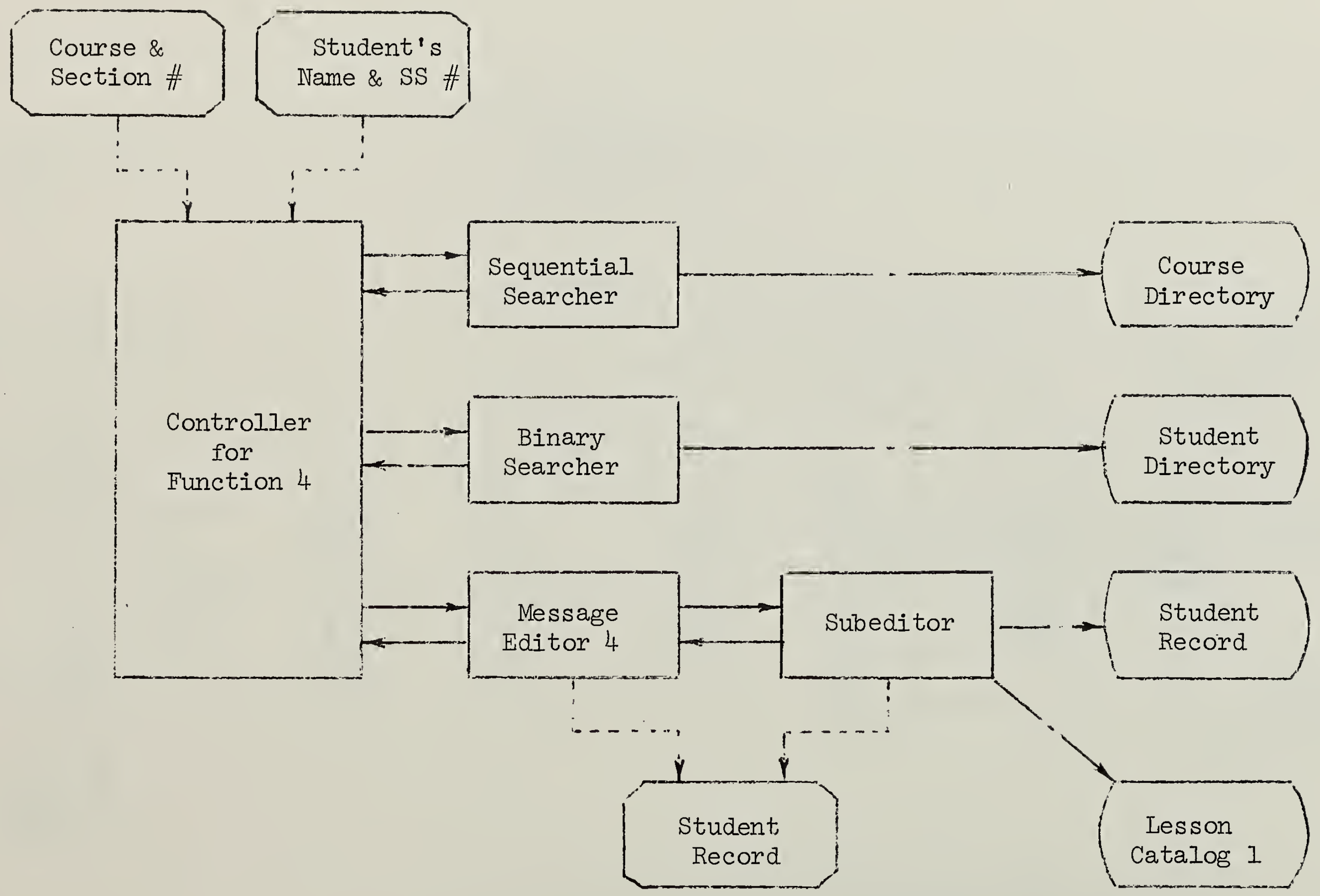

Figure 4.9 Structural Relationship between Modules - 4

for the specified course and section, and returns the "logical" location of the course and section in the Course Directory.

Then Controller 4 receives the student's name and social security number and joins the Binary Search module. The Binary Search module searches the specified (by PSDIRCT and NSTUDNT fields of the Course Directory) portion of the Student Directory for the specified social security number and returns the "logical" location of that number in the Student Directory. Next, Controller 4 joins Message Editor 4. Message Editor 4 generates the part of display image which is unique to the student record, obtains the location and length of the student record in Student Record from the Student Directory and Course Directory (PSRECRD and NLSREC) and joins the Subeditor which is shared with Message Editor 3. The Subeditor reads the specified portion of the Student Record and Lesson Catalog 1, and generates the rest of the display image.

## 2. Lexical Analyzer for Search Prescription ("lexi")

### 2.1 General

The Lexical Analyzer is joined by the Syntax Analyzer to get from the search prescription the next token to be analyzed.

### 2.2 Input

The input is a search prescription (explained in II.1.1) which is stored in the array "sprescr(1)" ~ "sprescr(lwnpres)". If the search prescription consists of less than 11 characters, then it occupies only the first word of the array, i.e., sprescr(1). If it consists of 11 ~ 20 characters, then it occupies the first two words of the array "sprescr(1)" and "sprescr(2)"; and so on.

## 2.3 Output

The output is a number stored in "crtoken" which designates one of the operators +, *, ', (,) and ;, if the number is positive, or the "logical" location of the operand (search word) in the search word table "searchw(1)" ~ "searchw(lwschw/2)", if the number is negative (Table 3). The search word table is an array each element of which consists of 2 words (= 20 characters). Thus the maximum length of a search word (a keyword) is 20 characters.

## 2.4 Functional Description

The Lexical Analyzer ("lexi") examines a search prescription stored in "sprescr(wnspres)" one character at a time. "lexi" has 4 states as shown in Figure 4.10. The operation depends on both the state and the current symbol (character) stored in "tcursym".

(a) State 0 (ready to get a new token)

If the current symbol in "tcursym" is one of the operators (+, *, ', (,) or ;), then "lexi" stores the code of the operator shown in Table 3, into "crtoken" and returns to the Syntax Analyzer. If the current symbol is a blank, "lexi" ignores it. If the current symbol is an upper case shift code, "lexi" goes to the State 1. If the current symbol is any other character, the symbol is assumed to be the first character of the new operand (search word or keyword), is stored into the first character position of the new location in the search word table ("searchw(wpsw)"), and goes to State 2.

| crtoken | MEANING |
|---|---|
| < 0 | The location of an operand (keyword) |
| 1 | The operator + (OR) |
| 2 | The operator * (AND) |
| 3 | The operator ' (NOT) |
| 4 | The operator ; |
| 5 | The operator ( |
| 6 | The operator ) |

Table 3. Code of Token

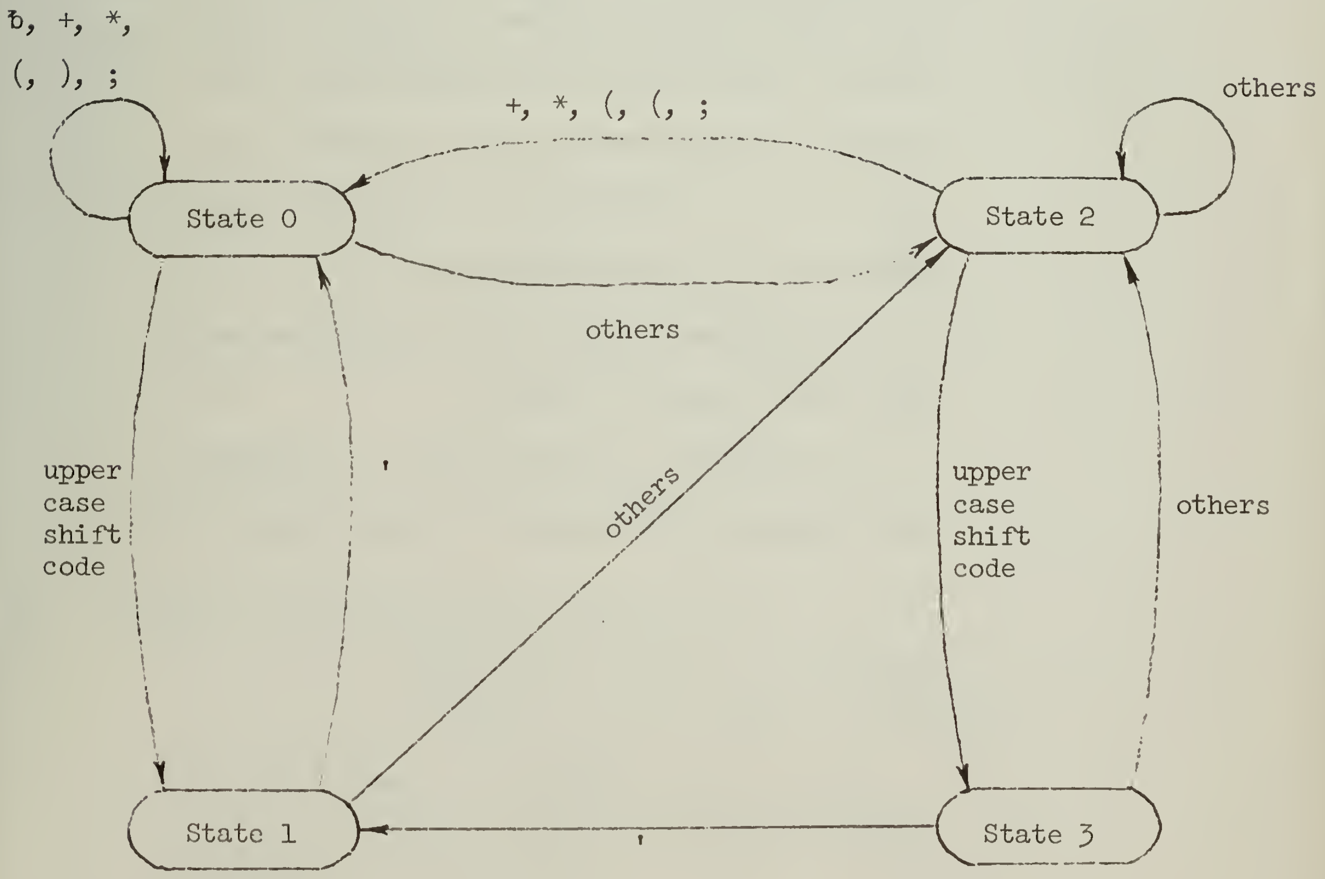

Figure 4.10 State Diagram of Lexical Analyzer

(b) State 1 (upper case character from State 0)

If the current symbol is ', then "lexi" stores 3 into "crtoken", goes to State 0, and returns to Syntax Analyzer.

Otherwise, the symbol is assumed to be the first character of the new operand, is stored into the first character position of the new location in the search word table, and goes to State 2.

(c) State 2 (getting an operand)

If the current symbol is one of the operators, "lexi" backs up one character position in the search prescription so that the current symbol appears again as the next symbol, negates the current location in the search word table and stores it into "crtoken", goes back to State 0, and returns to the Syntax Analyzer.

If the current symbol is the upper case shift code, "lexi" goes to State 3.

Other wise, the current symbol is assumed to be a character of the current operand, and is stored into the current character position of the current location in the search word table.

(d) State 3 (upper case from State 2)

If the current symbol is ', "lexi" backs up one character position in the search prescription so that ' appears again as the next symbol, negates the current location in the search word table and stores it into "crtoken", goes to State 1, and returns to the Syntax Analyzer.

Otherwise, "lexi" assumes the current symbol is a character of the current operand, stores it into the current character position of the current location in the search word table, and goes back to State 2.

## 3. Syntax and Semantics Analyzer of Search Prescription ("parser")

### 3.1 General

The Syntax and Semantics Analyzer ("parser") analyzes a search prescription according to the simple operator precedence grammar [6] described below, and outputs the result in the form of Polish postfix notation.

### 3.2 The Grammar of the Search Prescription

```
<Search Presc> ::= <Expression>;
<Expression>   ::= <Expression> + <Term> | <Term>
<Term>         ::= <Term> * <Factor> | <Factor>
<Factor>       ::= <Factor> ' | <Primary>
<Primary>      ::= (<Expression>) | <identifier>
```

### 3.3 Precedence Relations Between Operators

The meaning of the symbol in Table 4 is as follows:

$R > S$ -- R has precedence over S

$R = S$ -- R and S have the same precedence

$R < S$ -- S has precedence over R

The numbers 2, 3, ..., 9 indicate no relations.

| | + | * | ' | ; | ( | ) | i |
|---|---|---|---|---|---|---|---|
| + | > | < | < | > | < | > | < |
| * | > | > | < | > | < | > | < |
| ' | > | > | > | > | 2 | > | 3 |
| ; | < | < | < | = | < | 4 | < |
| ( | < | < | < | 5 | < | = | < |
| ) | > | > | > | > | 6 | > | 7 |
| i | > | > | > | > | 8 | > | 9 |

Table 4. Precedence Matrix

## 3.4 Input

The input to "parser" is a sequence of tokens extracted from a search prescription by "lexi" as described in the preceding section.

## 3.5 Output

The output of "parser" is a search prescription transformed into Polish postfix notation. This is stored in the array "postfix(1)" ~ "postfix(lpstfix)". Each element of the array contains either an operator or an operand. Operators are encoded as shown in Table 3. Operands in the postfix notation are the "logical" locations of the operands in the search word table.

## 3.6 Functional Description

The main frame of the "parser" consists of the following:

(a) Precedence Matrix "precdnc(1)" ~ "precdnc(49)"

(b) The current token "crtoken"

(c) Unreduced tokens in the syntax stack "pstack(1)" ~ "pstack(lpstack)"

(d) The parsed result in "postfix(1)" ~ "postfix(lpstfix)".

"parser" first joins the "lexi" and gets the current token. If the current token in "crtoken" is an operand, the contents in "crtoken" is saved in "crident" and the code designating identifier is assigned into "crtoken". If the current token is an operator, no operation is done in this stage.

Next, the precedence is examined between the top-most operator in the syntax stack "pstack(j)" and the current token in "crtoken". If the operator in the stack has the precedence over the current token, "parser" keeps examining the precedence between the next top-most operator and the current token until it finds the head of the prime phrase to be reduced, and reduces the prime phrase according to the grammar described in 3.2, producing the postfix notation appropriately. If the operator in the stack does not have the precedence over the current token, the contents in "crtoken" is stored at the top of the syntax stack "pstack(i)".

"parser" repeats these operations until it encounters ;.

In the precedence matrix "prednc(1)" ~ "prednc(49)", -1 denotes <, 0 denotes =, and 1 denotes >. Other numbers denote that no relation exists between the two operators.

## 4. Search Range Vector Calculator "calcsrv"

### 4.1 Search Range Vector

The Search Range Vector is a bit-string which represents a set of lessons. Each bit of the string corresponds to a lesson in the data base. In the current implementation of GUIDE-0, the Search Range Vector consists of 60 bits representing 60 lessons. For example, the following search range vector

$$\underbrace{\texttt{01100010 ...... 010}}_{\text{60-bit}}$$

represents the lessons #2, #3, #7 and #59. Lesson #2 means the lesson whose lesson number, i.e., the "logical" address of the lesson in Lesson Catalog 1, is 2.

## 4.2 Basic Function of "calcsrv"

The Search Range Vector Calculator "calcsrv" evaluates an expression which represents a search prescription in the form of Polish postfix notation (this expression is generated by the Parser). The result is expressed in the form of a search range vector, and is given to Message Editor 1.

## 4.3 Input

The input to "calcsrv" is a logical expression which represents a search prescription in the form of Polish postfix notation. The expression is generated by the Parser and is stored in the array "postfix(pointps)". See IV.3.5 for more detail.

## 4.4 Output

The output of "calcsrv" is a search range vector stored in an element of the array "tsrange(1)" ~ "tsrange(ltsrnge)". The location of the element where the search range vector is stored is given by "pstack(1)".

## 4.5 Functional Description of "calcsrv"

The Search Range Vector Calculator "calcsrv" is basically a conventional postfix expression interpreter. "calcsrv" scans the "postfix(pointps)". If it encounters an operand, it stores the operand into the stack "pstack(j)". If it encounters an operator, it executes the operation specified by the operator on the relevant operands which have been stored on the top locations of the stack "pstack", and then stores the result on the top of the stack.

There are two kinds of operands: one is a keyword stored in the search word table "searchw(1)" ~ "searchw(lwpsw)", and the other is a search range vector stored in the search range vector table "tsrange(1)" ~ "tsrange(ltsrnge)". In the "pstack", a keyword is represented by the negated "logical" address of the keyword in the search word table "searchw", while a search range vector is represented by

> the negated "logical" address of the search
> range vector in the search range vector
> table "tsrange" minus "lwpsw"

where "lwpsw" is the maximum number of keywords which can be stored in the search word table "searchw(1)" ~ "searchw(lwpsw)". Thus if $- \text{lwpsw} \leq \text{pstack}(j) \leq - 1$, then pstack(j) represents a keyword which is stored in "searchw(-pstack(j))", while if $\text{pstack}(j) < - \text{lwpsw}$, then pstack(j) represents a search range vector which is stored in "tsrange(-pstack(j)-lwpsw)".

Thus, before the operation is executed on the operands, the operands have to be checked to see whether they are keywords or search range vectors.

If the current operand is a keyword, i.e.,

$$- \text{lwpsw} \leq \text{pstack}(j) \leq - 1,$$

then the operand first has to be transformed to the corresponding search range vector. The search range vector for a keyword is given by the retrieval code attached to the keyword. Thus, if the operand is a keyword, "calcsrv" searches the keyword table for the keyword and obtains the retrieval code attached to it.

After all the operands relevant to the operation are transformed to search range vectors, the operation (AND, OR, NOT) is executed bit by bit on the operands (which are search range vectors). The result of the operation (which may be the operand of a further operation) is stored into the search range vector table "tsrange(q)". The location "q" of the result (actually -q-lwpsw) is stored on the top of the stack "pstack(j)".

Thus the final result of the calculation is put in the search range vector table "tsrange(-pstack(1)-lwpsw)" in the form of a search range vector which represents a set of lessons which match the given search prescription. The location of the final result in the search range vector is given by -pstack(1)-lwpsw.

## 5. Sequential Search Module "ssearch"

### 5.1 General

The Sequential Search Module "ssearch" is a general subroutine which searches specified locations of the common storage area in PLATO IV for a specified keyword of the specified length, and returns the "logical" address of the keyword.

### 5.2 Input (Parameters)

(a) "sdb" -- Start address of the file

The starting address of the file in the "common" storage area which is to be loaded and searched.

(b) "nwload" -- Physical length of the file

The number of physical words to be loaded and searched.

(c) "flength" -- Logical length of the file

The number of items in the file to be loaded and searched.

(d) "scom" -- Start address of common variable

The starting address of the common variables into which the file is loaded.

(e) "key(1)" ~ "key(2)" -- Keyword

The keyword to be searched for (up to 20 characters). The keywords of less than 20 characters are left aligned in "key(1)" and "key(2)".

(f) "kylengt" -- Length of keyword

The number of characters in the keyword.

(g) "subscrp" -- Expression

The expression to calculate the physical address of the specific field which is to be searched within the common variables. For example, suppose that a file, each element (item) of which consists of three physical words, is loaded into the common variables nc126 ~ nc185 (therefore, "scom" = 126, "nwload" = 185-(126-1) = 60, "flength" = "nwload"/3 = 60/3 = 20). Furthermore, suppose that the field to be searched is the second word of each item. Then the expression should be

3*(nl2-1)+126+1 = 3*nl2+124

where nl2(=i) contains the logical address of the items of the file. In other words, the expression maps the logical address (1, 2, 3, ..., 20) into the physical address (127, 130, 133, ..., 184).

(h) "count" -- Character count

The number of characters in the above expression.

## 5.3 Output

The output is the logical address, i, of the searched item in the file. If not found, a value of 0 is returned.

## 5.4 Functional Description

The specified file (by "sdb" and "nwload") is loaded into the specified common variables (by "scom"). Then the specified field (by "subscrp" and "kylengt") is searched for the keyword stored in "key(1)" and "key(2)" all through the file sequentially. If an item whose field matches the keyword is found, the logical address i of the item is returned. If not found, a value of 0 is returned.

## 6. Binary Search Module "bsearch"

### 6.1 General

The Binary Search Module "bsearch" is a general subroutine which searches the specified fields of the specified locations of the common storage area for a specified keyword of the specified length, and returns the "logical" address of the item whose specified field matches the keyword. If an item which matches the keyword cannot be found, a "would-be-address" is returned in negative form. The items are supposed to be sorted lexicographically by the specified field in order that the binary search[7] can be executed.

### 6.2 Input (Parameters)

In addition to the input to the Sequential Search Module "ssearch", the following input is necessary:

"bsmask" -- Mask pattern for the keyword

The mask pattern for the last "physical" word of the keyword. For example, if the keyword consists of 17 characters, the first 10 characters are contained in KEY(1) and the last 7 characters are contained in KEY(2). Thus, the "bsmask" should contain $00\underbrace{77777777777777}_{2x7}0000$, i.e., the 6-bit right shifted mask pattern for 7 characters. If the keyword consists of 6 characters, "bsmask" would be $00\underbrace{777777777777}_{2x6}000000$.

## 6.3 Output

The output is the logical address, i, of the searched item in the file whose specified field matches the keyword. If the item matching the keyword is not found, the "would-be-address" of the searched item in the file is returned in the negative form.

## 6.4 Functional Description

The specified (by "sdb" and "nwload") file is loaded into the specified (by "scom" and "nwload") common variables. The starting address for the binary search is calculated by "flength". Then the specified (by "subscrp", "kylengt" and "bsmask") field of the address is compared with the keyword. The unit of comparison is a maximum of 9 characters. For example, if the keyword consists of 16 characters, the first 9 characters are compared first. If matching occurs, then the next 7 characters are compared. The comparison is numerical in order to tell the next search location (forward or backward). This is the reason why the unit of comparison is not 10 characters (full word) but 9 characters and operands are right-shifted in order to avoid a possible negative value.

If the item whose field matches the keyword is found, the "logical" address i of the item is returned. If not found, the negated "would-be-address" is returned as the value of i.

## 7. Message Editors

Message Editors consist of 5 separate modules:

(a) Message Editor 1 "edtlsp"

(b) Message Editor 2 "edtdld"

(c) Message Editor 3 "edtdco"

(d) Message Editor 4 "edtdsr"

(e) Subeditor "edtdata"

The first four correspond to the four functions of GUIDE-0 while the last is a common subroutine for Message Editor 3 and 4.

## 7.1 Message Editor 1 "edtlsp"

Message Editor 1 "edtlsp" is the output message editor for the function 1 (display lessons which match search prescription). First it displays the headings on the top of the screen. Then it obtains the final result (search range vector) of the calculation done by "calcsrv", and displays the lesson names and abstracts of the lessons which are represented by the search range vector. Note that each bit of a search range vector corresponds to a lesson and the bit position shows the lesson number, i.e., the "logical" address of the lesson in Lesson Catalog 1. Thus, "edtlsp" scans the search range vector, detecting the bit positions which contain 1's, and displays the lesson name and abstract fields of the corresponding lessons in Lesson Catalog 1. If the number of lessons represented by the vector exceeds the number which can be displayed at one time on the screen, the lessons are paged and the control of turning pages are done by NEXT and BACK key. If the search range vector represents a null set, the editor displays a message which suggests that the user broadens the search range.

An example of the output is shown in Figure 4.11.

## 7.2 Message Editor 2 "edtdld"

Message Editor 2 "edtdld" is the output message editor for function 2 (display lesson descriptors). As explained in IV.1.3, the Binary Search module obtains the "logical" address of the specified lesson

```
  LESSON                  ABSTRACT

racetrack        Simulation Experiment

somaga           Software Management Game to Teach Programming

montecarlo       Area Calculation by Monte Carlo Method
```

Figure 4.11 Output Format for the Function 1

in Lesson Catalog 2. Then "edtdld" gets the lesson number of the lesson, i.e., the "logical" address of the lesson in Lesson Catalog 1, from the lesson number field "glesnn" in Lesson Catalog 2.

Now since "edtdld" knows both "logical" addresses of the lesson in Lesson Catalog 1 and 2, it displays the necessary fields of the lesson in Lesson Catalog 1 and 2 with the corresponding titles.

An example of the output is shown in Figure 4.12.

## 7.3 Message Editor 3 "edtdco"

Message Editor 3 "edtdco" is the output message editor for function 3 (display course outline). Since the file structures of the Course Outline and the Student Record are identical and also the output message formats for the function 3 and 4 are essentially the same, Message Editor 3 and Message Editor 4 use the common subroutine Subeditor "edtdata" for displaying the data in the Course Outline and the Student Record.

"edtdco" displays the headings which are unique to function 3. Then it obtains the "logical" pointer and the "logical" length of the course outline of the specified course from the Course Directory. After this, it calculates the starting "physical" address and the "physical" length of the course outline to be loaded. Note that the "logical" address of the course in the Course Directory is given by the Sequential Search module. Then the Subeditor "edtdata" is joined to display the data in the Course Outline.

An example of the output is shown in Figure 4.13.

## 7.4 Message Editor 4 "edtdsr"

Message Editor 4 "edtdsr" is the output message editor for function 4 (display student record). Like "edtdco", it first displays the

| | | | |
|---|---|---|---|
| LESSON NAME: | pl1do | Type: | exercise |
| ABSTRACT: | Introduction to PL/1 | | DO-statement |
| CATEGORY: | 2.1 , 3.63 | | |
| KEYWORDS: | iteration | | flow of control |
| | pl/1 | | |
| TIME REQUIRED: | 40 min. | | |
| PREREQUISITES: | | | |
| pl1data | Beginning Computer Science Lessons | | |
| pl1io | PL/1 Input/Output | | |
| pl1array | Introduction to PL/1 Arrays | | |
| SEQUELS: | | | |
| pl1if | PL/1 IF-THEN-ELSE Statements | | |

Figure 4.12 Output Format for the Function 2

```
   LESSON         TYPE       PERFORM EXPCTD    TIME REQRD     DATE

racetrack         game                           20 min.      9/10/73

plldata         exercise           70            40 min.      9/15/73

pllops          exercise           70            40 min.      9/25/73

pllarray        exercise           65            40 min.     10/10/73

plldo           exercise           70            40 min.     10/20/73

exam              exam             70            50 min.     10/25/73

pllif           exercise           65            60 min.      11/5/73
```

Figure 4.13 Output Format for the Function 3

headings which are unique to function 4, and then calculates the "physical" address and length of the student record which is to be displyaed. The "physical" address is deduced from the "logical" pointer to the student record, which is obtained from the "psrecrd" field of the Student Directory. The "physical" length comes from the "logical" length of the student record given by Controller 4. Then it joins the Subeditor "edtdata" to display the data in the student record.

An example of the output is shown in Figure 4.14.

### 7.5 Subeditor "edtdata"

Subeditor "edtdata" is the common subroutine which is used by both the Editor 3 and 4 to display the data stored in the Course Outline or the Student Record. "edtdata" first loads the specified part of the Course Outline or the Student Record and the Lesson Catalog 1. Note that both files have the identical file structure. Then "edtdata" displays the data in every field of the Course Outline or the Student Record except for the lesson number field. The lesson number is used to obtain the lesson name from Lesson Catalog 1.

## 8. Miscellaneous Modules

The following modules are usually deactivated and should be activated only for very special occasions such as the change of the specification of GUIDE-O itself or the introduction of the new modules to GUIDE-O system.

### 8.1 "inprecd" -- Initialize the Precedence Matrix

This module is used to initialize the precedence matrix "prednc(1)" ~ "prednc(49)" for the "parser". The numbers and their meanings are as follows on page 51.

```
  LESSON       TYPE      PERFORMANCE      TIME SPENT    DATE

racetrack      game                        30 min.      9/5/73

plldata      exercise        85            40 min.     9/10/73

pllops       exercise        65            30 min.     9/22/73

pllarray     exercise        80            55 min.     10/7/73

plldo        exercise                         min.       / /

exam           exam                           min.       / /

pllif        exercise                         min.       / /
```

Figure 4.14 Output Format for the Function 4

|  |  |  |
|---|---|---|
| -1 | -- | < |
| 0 | -- | = |
| 1 | -- | > |
| 2 ~ 9 | -- | no relation |

## 8.2 Setting the Experimental Data into the Data Base

The module names and their corresponding file names are listed in Table 5.

| MODULE | FILE |
|---|---|
| debug1 | Lesson Catalog 2 |
| debug3 | Keyword Table |
| debug6 | Course Directory |
| debug7 | Student Directory |
| dbgclg1 | Lesson Catalog 1 |
| dbgsco | Course Outline |
| dbgssr | Student Record |

Table 5. Data Base Initialization Modules

## 8.3 Displaying the Variables for Debugging Purpose

(a) "debug2"

Displays the keyword stored in the Search Word Table. Used as a subroutine for "debug4".

(b) "debug4"

Displays the contents of the Postfix and the Search Word Table by using "debugs" and "debug2". Used in Controller 1.

(c) "debugs"

Displays the contents of the Postfix. Used as a subroutine for debug4.

(d) "debug5"

Displays the search range vector in the Search Range Vector Table. Used in Controller 1.

# V. FUNCTIONAL SPECIFICATION OF THE DATA BASE EDITOR

GUIDE-O File Editor consists of two parts, Lesson Catalog Editor and Course Record Editor, corresponding to the structure of the files of GUIDE-O explained in the previous chapter.

In the following specification the necessary functions are listed in somewhat random fashion. These functions may be divided into subfunctions or be synthesized into more comprehensive functions to achieve the effective maintenance of the GUIDE-O files.

## 1. Lesson Catalog Editor

### 1.1 Addition of a New Lesson Name into Lesson Catalog

(a) Insert a new lesson name at the first available location in Lesson Catalog 1.

(b) Clear* the corresponding Subject Category and Keyword fields in Lesson Catalog 1.

(c) Insert the new lesson name into Lesson Catalog 2 in such a way that the resultant Lesson Catalog 2 is sorted lexicographically by lesson names.

(d) Set the corresponding "Lesson Number" field in Lesson Catalog 2 to the lesson number of the new lesson (this is the pointer to the new lesson in Lesson Catalog 1 or the "logical" location of the new lesson Lesson Catalog 1,

---

*If the field to be "cleared" is a character string, "clear" means to store "blank" characters (octal 55) into the field. Otherwise "clear" means to set the field to 0.

i.e., the lesson number is an integer between 1 and 60).

(e) Clear the corresponding "Time Required", "Relation to Other Lessons" and "Type" fields.

1.2 Replacement of Abstract

Replace an abstract in the Abstract field of Lesson Catalog 1 of the lesson specified (by lesson name).

1.3 Addition of a Subject Category Code

Add a subject category code into the Subject Category field of the specified lesson, if the field is not full.

1.4 Deletion of a Subject Category Code

Delete a specified subject category code in the Subject Category field of the specified lesson. "Delete" means to change the specified code into blank characters.

1.5 Replacement of a Subject Category Code

Replace the specified subject category code of the specified lesson with the specified subject category code (the combination of 1.3 and 1.4).

1.6 Addition of a Keyword

(a) Test whether or not the space is available for the addition of the specified keyword in the Keyword Field of the specified lesson in Lesson Catalog 1. If not available, no operation.

(b) If the space is available in Keyword field in Lesson Catalog 1, search the Keyword Table for the specified keyword. If the keyword is found, modify the corresponding retrieval code in Retrieval Code field of Keyword Table (set to 1 the bit which corresponds to the specified lesson).

If the keyword is not found and if space is available in the Keyword Table, insert the keyword into the Keyword Table so that the resultant Keyword Table is sorted lexicographically by keywords, set the corresponding retrieval code to all 0's but one bit which corresponds to the specified lesson, and modify all the keyword identification codes in the Keyword field of Lesson Catalog 1 which correspond to the keywords located after the newly inserted keyword in the Keyword Table (add one to all the codes).

If the space is not available, i.e., 255 keywords have already occupied Keyword Table, then no operation.

(c) If the specified keyword is found or inserted in Keyword Table, add the identification code of (the "logical" location of or the "logical" pointer to) the keyword (8-bit code representing 1 ~ 255) into Keyword field of the specified lesson in Lesson Catalog 1.

1.7 Deletion of a Keyword

(a) Search the Keyword Table for the specified keyword and obtain the corresponding identification code.

Note that the identification code is the logical location of the keyword in the Keyword Table and is not actually stored in the Keyword Table.

(b) Modify the corresponding retrieval code (set to 0 the bit which corresponds to the specified lesson). If the resultant retrieval code is all 0's, i.e., if no lessons are related to the keyword, delete the keyword from the Keyword Table, relocate all the keywords which are originally located after the deleted keyword, and modify all the identification codes of the keywords which are relocated in the Keyword field of Lesson Catalog 1.

(c) Delte the identification code of the keyword obtained in (a) from the Keyword field of the specified lesson in Lesson Catalog 1.

## 1.8 Replacement of a Keyword

Combination of 1.6 and 1.7.

## 1.9 Addition of a Relation to Other Lesson

Add the specified relation pair consisting of a relation code and a lesson number into the Relations to Other Lessons field of the specified lesson in Lesson Catalog 2, if the field is not full (i.e., the field contains less than 4 relation pairs).

## 1.10 Deletion of a Relation to Other Lesson

Delete the specified relation pair from the Relations to Other Lessons field of the specified lesson in Lesson Catalog 2.

1.11 Replacement of a Relation to Other Lesson

Combination of 1.9 and 1.10.

1.12 Replacement of Lesson Type

Replace the 4-bit lesson type code in Type field of the specified lesson in Lesson Catalog 2 with the 4-bit code of the specified lesson type.

1.13 Replacement of the Time Required

Replace the time required in Time Required field of the specified lesson in Lesson Catalog 2 with the specified time required.

2. Course Record Editor

2.1 Registration of a Course

Reserve spaces for a course outline and a student directory of the course.

(a) Search the Course Directory for the specified course and section. If found, then error.

(b) Insert the course and section number into Course and Section Number field at the first available space in the Course Directory.

(c) Reserve the specified amount (the maximum number of lessons) of space in Course Outline, and store the "logical" pointer to and the length of (i.e., the maximum number of lessons which can be contained in the course outline) the space into Pointer to Course Outline and Length of Course Outline fields in the Course Directory.

(d) Store the specified length of the student record (the maximum number of lessons which can be recorded in the student record) of the course into Length of Student Record field in the Course Directory. Note that the spaces for the student record are not reserved at this stage.

(e) Reserve the specified amount (the maximum number of students to be registered in the course) of space in Student Directory, clear Social Security Number field of the reserved space, and store the "logical" pointer and length of the space into Pointer to Student Directory and Length of Student Directory fields of the Course Directory.

(f) Set to 0 the Number of Students field of the Course Directory.

## 2.2 Deletion of a Course

Search Course Directory for the specified course and section number. If not found, no operation. If found, change the course and the section number in the Course and Section Number field to a string of blank characters.

## 2.3 Insertion of a Lesson into Course Outline

Insert a specified lesson (Lesson Number, Date and Performance Required) after the specified line in the course outline of the specified course. Note that the user doesn't specify the Lesson Number but Lesson Name. The editor must find out the lesson number of the specified lesson.

## 2.4 Deletion of a Lesson from Course Outline

Delete the lesson at the specified line of the course outline of the specified course, and relocate the lessons which are located after the deleted lesson.

2.5 Replacement of a Lesson in Course Outline

Replace the lesson at the specified line of the course outline of the specified lesson with the specified lesson (Lesson Number, Date and Performance).

2.6 Registration of a Student into a Course

(a) Search the student directory of the specified course for the specified social security number. If found, error.

(b) Insert the specified social security number and the student's name at the numerically ordered (by social security number) position in the student directory of the specified course.

(c) Reserve a specified (by Length of Student Record field of the specified course in Course Directory) amount of space in Student Record for the specified student and store the "logical" location of the reserved space into Pointer to Student Record field of the Student Directory.

(d) Copy lesson numbers in the course outline of the specified course into the corresponding field of the student record just reserved above.

If the student record has more space than the corresponding course outline, the rest of the space (Lesson Number field) must be cleared.

2.7 Deletion of a Student from a Course

(a) Search the student directory of the specified course for the specified social security number. If not found, no operation.

(b) Delete all the items (Social Security Number, Student Name, Pointer to Student Record) of the specified student from the student directory of the specified course, and relocate the items which are located after the deleted student.

## 2.8 Modification of "Date" and "Performance" of Student Record

This function must be activated by the system program of PLATO IV system, i.e., whenever a student terminates a lesson, this function is executed.

(a) Check the lesson type of the lesson the student just finished. If the lesson was an exam type and the lesson had been taken before by the same student, then do nothing. Otherwise,

(b) Modify Date, Performance and Time Spent field of the student record of the lesson in the student record of the student in the specified course.

## 2.9 Garbage Collection

Garbage collection is activated (called) if the space is not available when Course Record Editor tries to allocate the space for course outline, student directory or student record.

The garbage collection of course outline, student directory and student record requires the following two functions:

(a) Formation of storage map

Form the storage map which tells the current status of storage usage by scanning the Pointer to Course Outline and the Length of Course Outline fields of Course Directory for the garbage collection of Course Outline (Pointer to Student Directory and Length of Student Directory fields for Student Directory; Pointer to Student Directory, Length of Student Directory and Length of Student Record fields of Course Directory and Pointer to Student Record field of Student Directory for Student Record).

(b) Reallocation of storage

Reallocates all the spaces currently used to the top of the storage and make a big available space at the bottom of the storage.

## LIST OF REFERENCES

[1] Nievergelt, J. and Reingold, E. M., "Automating Introductory Computer Science Courses" in the proceedings of the 3rd ACM-SIGCSE Symposium on Computer Science Education (1973), 24-25.

[2] Alpert, D. and Bitzer, D. L., "Advances in computer based education", Science 167 (1970), 1582-1590.

[3] Pradels, J., "The GUIDE", (Report for Ph.D. Preliminary Examination), Department of Computer Science, University of Illinois at Urbana-Champaign, Urbana, Illinois.

[4] Shirer, D. and Sherwood, B., "aid1", PLATO IV lesson, Computer-based Education Research Laboratory, University of Illinois at Urbana-Champaign, Urbana, Illinois.

[5] Sherwood, B. et al., "aid2", PLATO IV lesson, Computer-based Education Research Laboratory, University of Illinois at Urbana-Champaign, Urbana, Illinois.

[6] Gries, D., "Compiler Construction for Digital Computers", Chapter 6, John Wiley & Sons, Inc., (1971), 122-132.

[7] Knuth, D. E., "The Art of Computer Programming", Volume 3, Chapter 6, Addison-Wesley (1969), 406-422.

# APPENDIX

## LISTING OF GUIDE-0

```
------------------------------------------------------------------------------------------- NOTES
 *
 *   +TO+: +GEORGE       6/24/73
 *      UTILITY AND LCATALOG ARE MOVED TO LESSON +.GUIDE+. WHICH
 *      HAS THE SAME CHANGE CODE AS +.GIODE0+.. PLEASE WORK ON
 *      +.GUIDE+. HEREAFTER.  ---  +SHINICHI

------------------------------------------------------------------------------------------- DEFINE1
 *

                                                                                     DEFINE1
UNIT     DEFINE1
COMMON   GUIDEDATA,GDATA,4186,NO LOAD
DEFINE   FIELDNM
         SAS=1788, SAR=2254
         SP1=12, SP2=18, SP3=24, SP4=9, SP5=6, SP6=4
         MASK1=07777, MASK2=077, MASK3=0777777, MASK4=0777
         MASK5=07777777777770000000, MASK6=0377, MASK7=017
         MASK8=07777777777777777777
         MASK9=07777777777777400000
         MASK10=07777777777777770000
         MASKB=0177, MASKC=037, MASKD=037777777777
         MASKF=07777777777777700000
 *
 *   DATA BASE FOR GUIDE0
 *     LESSON CATALOG1
         WPIC1=9    $$ NO OF WORDS PER LESSON
         SDC1=1501  $$ STARTING ADDRESS OF LESSON CATLOG1 IN
 *          THE COMMON STORAGE
         SC1=0    $$ STARTING ADDRESS OF LCATALOG1
         LESSNM1(I)=NC(WPIC1*(I-1)+SC1+1)
         ABSTRCT(I)=NC(WPIC1*(I-1)+SC1+2)
         LABSTRC=62    $$ MAX NO OF CHARACTERS OF ABSTRACT
         SUBJCAT(I)=NC(WPIC1*I+SC1-1) $CLS$ 12 $MASK$ MASK10
         NSUBJCT=2   $$ MAX NO OF S CATEGORY CODES IN SUBJCAT
         NDSUBJC=4   $$ MAX NO OF DIGITS OF SUBJ CATEGORY CODE
         KEYCODE(I)=NC(WPIC1*I+SC1)
         NKEYWRD=7   $$ MAX NO OF KEYWORD ID CODE IN KEYCODE
         NBKEYCD=8   $$ NO OF BITS OF KEYWORD ID CODE
         NICATLG=60  $$ MAX NO OF LESSONS IN CATALOG
         LC1=540     $$ NO OF WORDS OF LCATALOG1
 *     LESSON CATALOG2
         SDC2=2041   $$ START ADDRESS OF LESSON CATLOG2 IN THE
 *          COMMON STORAGE
         SCATLG2=SC1+LC1
         NWLCLG2=2   $$ NO OF WORDS PER LESSON OF LCATALOG2
         LESSNM2(I)=NC(NWLCLG2*I+SCATLG2-1)    $$ LESSON NAME
         GLESSNN(I)=NC(NWLCLG2*I+SCATLG2) $CLS$ 6 $MASK$MASK2
 *          LESSON NO
         GTIME(I)=NC(NWLCLG2*I+SCATLG2) $CLS$ 12 $MASK$ MASK2
         RELATN(I)=NC(NWLCLG2*I+SCATLG2) $CLS$ 12 $MASK$MASK9
         GTYPE(I)=NC(NWLCLG2*I+SCATLG2) $MASK$ MASK6
         NICTLG2=11    $$ NO OF ITEMS IN LCATALOG2
 *   THE ABOVE VALUE IS FOR DEBUG(SHOULD BE 60)
```

```
LESSON GUIDED         AT      08.59.28.     ON      06/24/73

        LCATLG2=120
•    KEYWORD TABLE
        SDKY=2161   $$ START ADDRESS IN THE COMMON STORAGE
        SKEYWDT=660, WPKKT=3
        KEYWORD(I)=NC(3*I+SKEYWDT-2)  $$ KEYWORD TABLE
        LKEYWRD=20     $$ KEY WORD LENGTH(NO OF CHARACTERS)
        RETCODE(I)=NC(3*I+SKEYWDT)     $$ RETRIEVAL CODE
        NIKYWDT=20   $$ NO OF ITEMS IN KEYWORD TABLE
•  THE ABOVE VALUE IS FOR DEBUG(SHOULD BE 256)
        LKEYWDT=768

------------------------------------------------------------------------------------------------------------------ DEFINE2
•
•    COURCE DIRECTORY
        SDCD=1  $$ START ADDRESS IN THE COMMON STORAGE
        SCDIRCT=110, NWPLC=3
        COURCEN(I)=NC(NWPLC*I+SCDIRCT-2)  $$ COURCE NUMBER
        LCOURCN=9   $$ NO OF CHARACTERS OF A COURCE NAME
        SECCODE(I)=NC(NWPLC*I+SCDIRCT-1)
        PCOUTLN(I)=NC(NWPLC*I+SCDIRCT) $CLS$ SP1 $MASK$MASK1
        LCOUTLN(I)=NC(NWPLC*I+SCDIRCT) $CLS$ SP2 $MASK$MASK2
        LSPECRD(I)=NC(NWPLC*I+SCDIRCT) $CLS$ SP3 $MASK$MASK2
        PSDIRCT(I)=NC(NWPLC*I+SCDIRCT) $ARS$ SP2 $MASK$MASK3
        LSDIRCT(I)=NC(NWPLC*I+SCDIRCT) $ARS$ SP4 $MASK$MASK4
        NSTUDNT(I)=NC(NWPLC*I+SCDIRCT) $MASK$ MASK4
        LCDIRCT=30, NICDIRC=10
•
•    COURCE OUTLINE
        SCOUTLN=3001  $$ START ADDRESS IN THE COMMON STORAGE
        SAD=SCDIRCT+LCDIRCT
        OLESSNN(I)=NC(I+SAD) $CLS$ SP5 $MASK$ MASK2
        ODATE(I)   =NC(I+SAD) $CLS$ SP5 $MASK$ MASK5
        OTIME(I)   =NC(I+SAD) $ARS$ SP1 $MASK$ MASK2
        OPERFRM(I)=NC(I+SAD) $ARS$ SP6 $MASK$ MASK6
        OTYPE(I)   =NC(I+SAD) $MASK$ MASK7
        LCO=150
•    STUDENT DIRECTORY
        SSDIRCT=31  $$ START ADDRESS IN THE COMMON STORAGE
        WPS=3
        SOCSECN(I)=NC(WPS*I-2)    $$ SOCIAL SECR NO
        SNAME(I)   =NC(WPS*I-1)
        PSRECRD(I)=NC(WPS*I) $MASK$ MASK3
        NSDIRCT=450
        OLESSNM(I)=NC(WPIC1*(I-1)+291)
•       TEMPORARY USE FOR COURCE OUTLINE DISPLAY
•    COURCE OUTLINE AND STUDENT RECORD
        SSRECRD=481  $$ START ADDRESS OF THE STUDENT RECORD
•         IN THE COMMON STORAGE
        WPLC=1
        CLESSNN(I)=NC(I+LC1) $CLS$ SP5 $MASK$ MASK2
        YEAR(I)=NC(I+LC1) $ARS$ 9 $MASK$ MASK8
        MONTH(I)=NC(I+LC1) $ARS$ 5 $MASK$ MASK7
        DAY(I)=NC(I+LC1) $MASK$ MASKC
        CTYPE(I)=NC(I+LC1) $CLS$ 12 $MASK$ MASK2
        PERFORM(I)=NC(I+LC1) $ARS$ 16 $MASK$ MASK6
```

```
 *
 *
          SECTLG2=2       $$ SIZE OF EACH ELEMENT OF LCATLG2
          SACTLG2=643     $$ STARTING ADDRESS OF LCATLG2

---------------------------------------------------------------------------------------------------------------- DEFINE3
 *
 *
          SPSPRES=542     $$ STARTING WORD OF SEARCH PRESCRIPT
          SPRESCR(I)=NC(I+SPSPRES)    $$ SEARCH PRESCRIPTION
          LWNPRES=20      $$ WORD NO OF SEARCH PRESCRIPTION
          SPSCHWD=SPSPRES+LWNPRES
          SEARCHW(I)=NC(2*I+SPSCHWD-1) $$ SEARCH WORD TABLE
          LWSCHWD=40
          SPPRECD=SPSCHWD+LWSCHWD
          PRECONC(I)=NC(I+SPPRECD)  $$ PRECIDENCE REL MATRIX
          LPRECD=49       $$ THE SIZE OF PRECIDENCE MATRIX
          SPTFIX=SPPRECD+LPRECD
          POSTFIX(I)=NC(I+SPTFIX)  $$ PARSED SEARCH PRESCRPT
          LPSTFIX=20
          STSRNGE=SPTFIX+LPSTFIX
          TSRANGE(I)=NC(I+STSRNGE)  $$ SEARCH RANGE TABLE
          LTSRNGE=20
          SREL=STSRNGE+LTSRNGE+1
          SCRELCM=SREL+3001
          CRELATN(I)=NC(I+SREL+500)
          LREL=16
          SLTYPE=LC1+LCATLG2+LKEYWDT+1
          SDLT=SCRELCM+LREL
          LTYPE(I)=NC(I+SLTYPE)  $$ LESSON TYPE DECODER
          LLTYPE=16
 *          LTYPE(0) -- LTYPE(LLTYPE-1)
          PSTACK(I)=N(I+50)    $$ SYNTAX STACK(N51-N60)
          LPSTACK=10

---------------------------------------------------------------------------------------------------------------- DEFINE4
 *
          KEY(I)=N(I)
          KYLENGT=N3      $$ KEY LENGTH(NO OF CHARACTERS)
          SUBSCRP=N4      $$ EXPRESSION TO COMPUTE ACTUAL ADDRS
          COUNT=N10       $$ LENGTH OF THE EXPRESSION(NO OF CH)
          CMPILED=N16     $$ ADDRESS OF THE COMPILED CODE
          FLENGTH=N11     $$ SIZE OF THE FILE TO BE SEARCHED
          HSMASK=N21
          STARTPS=N41     $$ START POSITION OF BINARY SEARCH
          K=N13           $$ COMPUTED VALUE OF ACTUAL SUBSCRIPT
          L=N14           $$ INCREMENT FOR BINARY SEARCH
          P=N15           $$ WORD NO IN THE FIELD OF FILE
          LESNAME=N17     $$ LESSON NAME TO BE SEARCHED
          TKYLENG=N18     $$ REMAINING LENGTH OF KEY
          CURITEM=N19     $$ CURRENT ITEM TO BE TESTED
          CURKEY=N20      $$ PARTIAL KEY TO BE COMPARED WITH
          TEMP=N70, TEMP2=N69, TEMP3=N68
          CRTOKEN=N23     $$ CURRENT TOKEN
          CRIDENT=N24     $$ CURRENT IDENTIFIER
```

```
*
*  INSTRUCTION DECODE
                                                                                IDECODE
UNIT     IDECODE
AT       505
WRITE    *WHICH OF THE FOLLOWING SERVICES DO YOU WANT*/
AT       708
WRITE    1) SEARCH FOR AND DISPLAY LESSONS WHICH MATCH THE
            SEARCH PRESCRIPTION
         2) DISPLAY LESSON DESCRIPTORS
         3) DISPLAY COURSE OUTLINE
         4) DISPLAY STUDENT RECORD
AT       2005
WRITE    *TYPE-IN THE NUMBER.
ARROW    2205
LONG     1
STORE    INSTRCT
OK
JUMP     INSTRCT-2, IDCLSP, IDCOLD, IDCOCO, IDCDSR
STOP
GOTO     INSTRCT=*,ASR        *,, X, ERR*/*/*/*/*/
*****
START
*
*
*  CONTROLLER 1
                                                                                IDCLSP
UNIT     IDCLSP
HELP     HSPRESC
DATA     DKEYTBL
AT       505
WRITE    *TYPE-IN A SEARCH PRESCRIPTION (A LOGICAL EXPRESSION
         OF KEYWORDS AND/OR PHRASES FOLLOWED BY A SEMICOLON).
AT       2505
WRITE    *H*E*L*P AVAILABLE.
         *D*A*T*A TO SEE THE AVAILABLE KEYWORDS.
ARROW    405
STOREA   SPRESCP(1),LSPRSCR
OK
UNLOADC  SPRESCP(1),SPSPRES,3001, LWNPRES
JOIN     PARSER
*  PARSE THE GIVEN SEARCH PR3ESCRIPTION
*
BREAK
STOP
JOIN     DEBUG4
*  DISPLAY POSTFIX(PARSED RESULT)
PAUSE
AT       2205
WRITE    INTERMEDIATE RESULTS
START
JOIN     CALCSRV
*  CALCULATE SEARCH RANGE VECTOR
```

```
      LESSON GUIDED        AT      08.59.28.     ON      06/24/73

*
STOP
JOIN     DEBUGS
*  DISPLAY THE FINAL SEARCH RANGE VECTOR
PAUSE
START
BREAK
ERASE
JOIN     EDTLSP
*  EDIT AND DISPLAY THE OUTPUT MESSAGE
PAUSE
JUMP     IDECODE
*
*  EXPLANATION OF THE SEARCH PRESCRIPTION(+H+E+L+P UNIT)

                                                                        HSPRESC
UNIT     HSPRESC
DATA     DKEYTBL
AT       307
WRITE    +THE SEARCH PRESCRIPTION IS A LOGICAL EXPRESSION OF
         KEYWORDS AND/OR PHRASES FOLLOWED BY A SEMICOLON.
         +THE ALLOWED LOGICAL OPERATIONS ARE +A+N+D(*), +O+R(+) AND
         +N+O+T(+7).  +OPERATORS AND OPERANDS MAY BE SEPERATED BY
         BLANKS, AND PARENTHETICAL EXPRESSION NESTED TO ANY
         LEVEL IS ALLOWED.
         +FOR EXAMPLE, SUPPOSE YOU WANT TO STUDY ABOUT DATA
         STRUCTURES OTHER THAN ARRAYS, IN ANY LANGUAGES OTHER
         THAN +P+L/1 AND +A+L+G+O+L, +THEN THE SEARCH PRESCRIPTION
         COULD BE AS FOLLOWS+!

             DATA STRUCTURE*ARRAY+7*(PL/1+ALGOL)+7!

         +BE SURE NOT TO FORGET A SEMICOLON.
AT       2507
WRITE    +D+A+T+A TO SEE THE AVAILABLE KEYWORDS.
END
*
*  LISTING OF AVAILABLE KEYWORDS

                                                                        DKEYTBL
UNIT     DKEYTBL
LOADC    KEYWORD(1), SDKY, LKEYWDT
CALC     I + 0
         K + 105
                                                                  NXTKEYW
ENTRY    NXTKEYW
CALC     K + K+100
GOTO     K>3000, X, IPLUS1
PAUSE
ERASE
CALC     K + 205
                                                                  IPLUS1
ENTRY    IPLUS1
CALC     I + I+1
GOTO     I>NTKYWDT, DKEXIT, X
```

```
AT       *
SHOWA    KEYWORD(1). 20
GOTO     UATKEYW
                                                                        OKEXIT
ENTRY    OKEXIT
END

------------------------------------------------------------------------------------ CONTROL
*
*  CONTROLLER 2

                                                                        IDCOLD
UNIT     IDCOLD
AT       905
WRITE    *WHAT LESSON DO YOU WANT*/
ARROW    1105
STOREA   KEY(1).10
OK
*  SET PARAMETERS FOR RSEARCH(SEE *.RSEARCH*. FOR THE MEANING
*  OF EACH PARAMETER)
CALC     KYLENGT * 10
         HSMASK * 07777777777777777777
         FLENGTH * NICTLG2
         CMPILED * 0
         STARTPS * 1
         SCOM*SCATLG2+1
         SDR * SOC2
         NWLOAD * LCATLG2
PACK     COUNT. SUBSCRP. 2*N12+539      << 1=N12
JOIN     HSEARCH
*  SEARCH THE LESSON CATALOG 2 FOR THE SPECIFIED LESSON
ERASE
JOIN     EDITOLD
*  EDIT AND DISPLAY THE OUTPUT MESSAGE
JUMP     IDECODE
*
*
*  CONTROLLER 3

                                                                        IDCOCO
UNIT     IDCOCO
AT       905
WRITE    *TYPE-IN THE COURSE AND SECTION NUMBER(EG. CS105A1).
ARROW    1105
STOREA   KEY(1).1 COURCN
OK
JOIN     SPCDIRC
*  SET PARAMETERS FOR SSEARCH
JOIN     SSEARCH
*  SEARCH THE COURSE DIRECTORY FOR THE SPECIFIED COURSE AND
*  SECTION
STOP
AT       1505
WRITE    I =
SHOW     I
```

```
*   SHOW THE SEARCH RESULT. IF NEGATIVE, NOT FOUND.
PAUSE
START
BREAK
ERASE
JOIN     EDTDCO
*   EDIT AND DISPLAY THE OUTPUT MESSAGE
PAUSE
JUMP     IDECODE
*
*   CONTROLLER 4


UNIT     IDCDSR                                                                          IDCDSR
AT       505
WRITE    +TYPE-IN THE COURSE AND SECTION NUMBER YOU ARE
         ENROLLED IN(EG. CS101E1).
ARROW    705
STOREA   KEY(1).LCOURCN
OK
JOIN     SPCDIRC
*   SET THE PARAMETERS FOR SSEARCH
JOIN     SSEARCH
*   SEARCH THE COURSE DIRECTORY FOR THE SPECIFIED COURSE AND
*   SECTION
GOTO     I<0.ERNOCRS.X
STOP
*   DEBUG
AT       1505
WRITE    I =
SHOW     I
*   SHOW THE SEARCH RESULT
PAUSE
START
BREAK
AT       1505
ERASE    1*30
LOADC    COURCEN(1).SDCD.LCDIRCT
*   LOAD THE COURSE DIRECTORY
*   SET THE PARAMETERS FOR THE BSEARCH
CALC     STARTPS → PSDIRCT(I)
         FLENGTH → NSTUDNT(I)
         SDR → SSDIRCT+(PSDIRCT(I)-1)*WPS
         NWLOAD → FLENGTH*WPS
         NLSREC → LSPECRD(I)
         CMPILED → 0
         SCOM → 1
PACK     COUNT.SUBSCRP.3*N12-2
AT       1305
WRITE    +TYPE-IN YOUR NAME(EG. FONDA,JANE)
ARROW    1505
STOREA   STDNAME.LSNAME
OK
AT       1705
WRITE    +TYPE-IN SOCIAL SECURITY NUMBER(EG. 346520078).
```

```
ARROW    1205
STUDFA   KEY(1),9
OK
CALC     KYLENGT ⇐ 2
         HSMASK ⇐ MASKB
JOIN     HSEARCH
*  SEARCH THE STUDENT DIRECTORY FOR THE SPECIFIED SOCIAL
*  SECURITY NO.
*
STOP
AT       2105
WRITE    I =
SHOW     I
*  SHOW THE SEARCH RESULT
PAUSE
START
BREAK
ERASE
JOIN     EDTDSR
*  EDIT AND DISPLAY THE OUTPUT MESSAGE
PAUSE
JUMP     IDECODE
*
                                                                         ERNOCRS
ENTRY    ERNOCRS
AT       1005
WRITE    +THE COURSE IS NOT INCLUDED IN OUR DATA BASE.
PAUSE
JUMP     IDECODE
*
-------------------------------------------------------------------------------------- SLPRFSC
*
*  SEARCH LESSONS BY KEYWORDS
                                                                                CALCSRV
UNIT     CALCSRV
CALC     POINTPS ⇐ 1 ⇐ 0 ⇐ 0
                                                                         INCRP
ENTRY    INCRP
LOADC    POSTFIX(1), SPTFIX+300, LPSTFIX+LTSRNGE
CALC     POINTPS ⇐ POINTPS + 1
GOTO     POSTFIX(POINTPS),X,EILGINF,BOP,BOP,OPRND2,OPRND2
CALC     J ⇐ J+1
GOTO     J>LPSTACK, EPSTOVF, X
CALC     PSTACK(J) ⇐ POSTFIX(POINTPS)
GOTO     INCRP
                                                                         BOP
ENTRY    BOP
GOTO     PSTACK(J-1)+LWPSW < 0, ISRNGE1, X
*        SEARCH KEYWORD
CALC     TEMP ⇐ J-1
BREAK
JOIN     SETPARM
JOIN     HSEARCH
```

```
GOTO    I>0, X, NOKEY
LOADC   KEYWORD(1), SDKY, LKEYWDT
CALC    SRANGE1 → RETCODE(I)
GOTO    OPRND2
*       OPERAND IS A SEARCH RANGE VECTOR
                                                                  ISRNGE1
ENTRY   ISRNGE1
CALC    SRANGE1 → TSRANGE(-PSTACK(J-1)-LWPSW)
                                                                  OPRND2
ENTRY   OPRND2
GOTO    PSTACK(J)+LWPSW<0, ISRNGE2, X
*       SEARCH KEYWORD
CALC    TEMP → J
BREAK
JOIN    SETPARM
JOIN    HSEARCH
GOTO    I>0, X, NOKEY
LOADC   KEYWORD(1), SDKY, LKEYWDT
CALC    SRANGE2 → RETCODE(I)
GOTO    INCRQ
                                                                  NOKEY
ENTRY   NOKEY
DATA    DKEYTBL
AT      2501
ERASE   120
AT      2505
WRITE   ↑AT LEAST ONE OF THE KEYWORDS SPECIFIED IS NOT
        INCLUDED IN THE ↑KEYWORD ↑TABLE.
PAUSE
JUMP    IDCLSP
                                                                  ISRNGE2
ENTRY   ISRNGE2
LOADC   TSRANGE(1), STSRNGE+3001, LTSRNGE
CALC    SRANGE2 → TSRANGE(-PSTACK(J)-LWPSW)
                                                                  INCRQ
ENTRY   INCRQ
LOADC   POSTFIX(1), SPTFIX+3001, LPSTFIX+LTSRNGE
CALC    Q → Q+1
GOTO    Q>LTSRNGE, ETSROVF, X
GOTO    POSTFIX(POINTPS)-2, X, AND, NEG, SCOLN
*  ↑,OR↑,
CALC    TSRANGE(Q) → SRANGE1 $UNION$ SRANGE2
GOTO    DECRJ
                                                                  AND
ENTRY   AND
CALC    TSRANGE(Q) → SRANGE1 $MASK$ SRANGE2
                                                                  DECRJ
ENTRY   DECRJ
CALC    J → J-1
GOTO    STACKQ
                                                                  NEG
ENTRY   NEG
CALC    TSRANGE(Q) → -SRANGE2  $$ COMPLEMENT
                                                                  STACKQ
ENTRY   STACKQ
```

```
CALC     PSTACK(J) → -Q-LWPSW
STOP
SHOW     PSTACK(J)
SHOWO    TSRANGE(-PSTACK(J)-LWPSW)
START
UNLOADC  TSRANGE(Q), W.STSRNGE+3000, 1
GOTO     INCPP
                                                                   SCOLN
ENTRY    SCOLN
GOTO     J=1, SKEXIT, X
AT       2505
WRITE    *ERROR*, *ILLEGAL POSTFIX FORMAT.
GOTO     SKEXIT
                                                                   EPSTOVF
ENTRY    EPSTOVF
AT       2505
WRITE    *ERROR*, PSTACK OVERFLOW DURING SEARCH PROCESS.
GOTO     SKEXIT
                                                                   ETSROVF
ENTRY    ETSROVF
AT       2505
WRITE    *ERROR*, TSRANGE OVERFLOW.
                                                                   SKEXIT
ENTRY    SKEXIT
PAUSE
CALC     TSRANGE(Q) → SRANGE2
         PSTACK(J) → -Q-LWPSW
UNLOADC  TSRANGE(1), STSRNGE+3001, LTSRNGE
EXIT     1
                                                                   EILGINF
ENTRY    EILGINF
AT       2505
WRITE    *ERROR*, *ILLEGAL INFORMATION IN POSTFIX, *IGNORED.
GOTO     INCRR
                                                                   LSRANGE
ENTRY    LSRANGE

------------------------------------------------------------------------------ CALCSRV2
*

                                                                        SETPARM
UNIT     SETPARM
LOADC    SEARCHW(1), SPSCHWD+3001, LWSCHWD
MOVE     SEARCHW(-PSTACK(TEMP)), 1, KEY(1), 1, 20
CALC     KYLENGT → LKEYWRD
         BSMASK  → MASKB
         FLENGTH → NTKYWDT
         CMPILED → 0
         STARTPS → 1
         SCOM → SKEYWDT+1
         SDB → SDKY
         NWLOAD → LKEYWDT
PACK     COUNT,SUBSCRP,3*N12+658
EXIT     1
*
```

```
                                                                        SPCDIRC
UNIT    SPCDIRC
PACK    COUNT,SUBSCRP,3*N12+108
CALC    FLENGTH ⇒ NICDIRC
        SDR ⇒ SDCD
        KYLENGT ⇒ LCOURCN
        FILE ⇒ 4
        CMPILED ⇒ 0
EXIT    1

---------------------------------------------------------------------------- SEARCH
*
*  --BINARY SEARCH SUBROUTINE--
*
*  SET THE FOLLOWING PARAMETERS BEFORE JOINING BSEARCH.
*  THE EXAMLE OF THE USAGE OF THE ROUTINE CAN BE FOUND IN
*  THE UNIT -INCOLD-(IN BLOCK -IDECODE-)
*  PARAMETERS:
*    KEY(I)  --- THE ITEM TO BE SEARCHED
*    KYLENGT --- NO OF CHARS OF KEY
*    BSMASK  --- MASK PATTERN FOR THE LAST WORD OF THE KEY
*                EG. IF THE KEY CONSISTS OF 17 CHARACTERS
*                INCLUDING BLANKS, BSMASK SHOULD CONTAIN
*                0007777777777777770000(NOTE THAT RIGHT SHIFT
*                ED 6-BIT. NOT 07777777777777000000.)
*    SDR  ------ START ADDRESS OF THE FILE TO BE SEARCHED
*                IN THE COMMON STORAGE(1 -- 4186)
*    SCOM   ---- START ADDRESS OF THE FILE IN THE COMMON
*                VARABLES(NC1 -- NC1500)
*    FLENGTH --- THE LENGTH OF THE FILE TO BE SEARCHED
*    NWLOAD  --- NO OF PHYSICAL WORDS IN THE FILE
*    SUBSCRP --- THE STRING FOR CALCULATION OF THE ACTUAL
*                SUBSCRIPT OF THE FILE
*    COUNT   --- NO OF CHARS OF SUBSCRP STRING

                                                                        BSEARCH
UNIT    BSEARCH
LOADC   NC(SCOM),SDR,NWLOAD
                                                                  CSITEM
ENTRY   CSITEM
*  CALCULATE STARTING ITEM NO
CALC    TEMP ⇒ FLENGTH
        L ⇒ 1
30      BRANCH TEMP>1, 10, 20
10      L ⇒ 2*L
        TEMP ⇒ TEMP/2
        BRANCH 30
20      I ⇒ L ⇒ L/2
                                                                  CMPT
ENTRY   CMPT
COMPUTE K, SUBSCRP, COUNT, CMPILED
*    SUBSCRP --- STRING TO BE COMPILED(1ST WORD)
*    COUNT ----- NO OF CHARS OR THE STRING
*    CMPILED --- POINTER TO COMPILED CODE
```

```
CALC    P ⇒ -
        TKYLENG ⇒ KYLENGT
05      P ⇒ P+1
        TKYLENG ⇒ TKYLENG-9
        BRANCH TKYLENG>0, 10, 20
10      CURITEM ⇒ NC(K+P-1) $ARS$ 6
        CURITEM ⇒ CURITEM $MASK$ 0777777777777777777
        CURKEY ⇒ KEY(P) $ARS$ 6 $MASK$ 0777777777777777777
        BRANCH 30
20      CURITEM ⇒ NC(K+P-1) $ARS$ 6 $MASK$ BSMASK
        CURKEY ⇒ KEY(P) $ARS$ 6 $MASK$ BSMASK
30      BRANCH CURITEM-CURKEY, 40, 50, 60
50      BRANCH TKYLENG>0, 70, 85
70      TKYLENG ⇒ TKYLENG-1
        CURITEM ⇒ NC(K+P-1) $MASK$ 07
        CURKEY ⇒ KEY(P) $MASK$ 07
        BRANCH CURITEM-CURKEY, 40, 90, 60
90      BRANCH TKYLENG>0, 05, 85
40      BRANCH L>1, 45, 73
45      L ⇒ L/2
        I ⇒ I+L
        BRANCH I>FLENGTH, 60, 75
60      BRANCH L>1, X, 80
65      L ⇒ L/2
        I ⇒ I-L
        BRANCH I>FLENGTH, 60, 75
73      BRANCH I-FLENGTH+1, 78, X, 78
        I ⇒ FLENGTH
75      TEMP ⇒ -1
        BRANCH 0
78      I ⇒ I+1
80      I ⇒ -I
85      TEMP ⇒ 0
        BRANCH 0
GOTO    TEMP, CMPT, X
EXIT    1
                                                                                ENOFILE
ENTRY   ENOFILE
AT      3005
WRITE   BINARY SEARCH CANNOT BE DONE ON THE SPECIFIED FILE.
CALC    I ⇒ 0
EXIT    1

------------------------------------------------------------------------------------------------ PARSER
*
*  SEQUENCIAL SEARCH SUBROUTINE
*
                                                                                        SSEARCH
UNIT    SSEARCH
GOTO    FILE=4,X
LOADC   COURCEN(1),SCDIRCT+1501,LCDIRCT
LOADC   COURCEN(1),SDR,LCDIRCT
*  LOAD COURCE DIRECTORY
CALC    I ⇒ 0
```

```
                                                                             NEXTITM
ENTRY    NEXTITM
CALC     I → I+1
GOTO     I>FLENGTH,NFOUND,X
COMPUTE K,SUBSCRP,COUNT,CMPILED
*  COMPUTE PHYSICAL  SUBSCRIPT
SEARCH   KEY(1),KYLENGT,NC(K),1,TEMP
GOTO     TEMP,NEXTITM,X
EXIT     1
                                                                             NFOUND
ENTRY    NFOUND
CALC     I→0
EXIT     1
*
*
*  SYNTAX AND SEMANTICS ANALYSER FOR SEARCH PRESCRIPTION
*  UNDER OPERATOR PRECIDENCE GRAMMAR
*

                                                                                    PARSER
UNIT     PARSER
DO       CLSCHWD, I→1,20
CALC     PSTACK(1) → DELIMIT
         I → 1
         STATE → POINTPS → 0
         WNSPRES → 0  $$ INIT WD NO OF SEARCH PRESCRPT
         CPSP → 10
         CPSW → WPSW → 1  $$ INIT CHAR, WORD POS OF SEARCH WD
                                                                             NEWTOKEN
ENTRY    NEWTOKEN
JOIN     LEXI
LOADC    PRECDNC(1),SPPRECD+3001,69
CALC     BRANCH CRTOKEN<0, X, 10
         CRIDENT → CRTOKEN
         CRTOKEN → CDIDENT
10       BRANCH PSTACK(I)>0, X, 20
         J → I
         BRANCH 30
20       J → I-1
30       TEMP → (PSTACK(J)-1)*NOFPROW + CRTOKEN
         BRANCH PRECDNC(TEMP), X, X, 40, 91, 92
         I → I+1
         PSTACK(I) → CRTOKEN
         FCPARSE → -1
         BRANCH 0
40       Q → PSTACK(J)
         J → J-1
         BRANCH PSTACK(J)>0, 50, X
         J → J-1
50       TEMP → (PSTACK(J)-1)*NOFPROW + Q
*        CALCULATE THE ARGUMENT OF THE PRECIDENCE RELATION
*        MATRIX
         BRANCH PRECDNC(TEMP), X, 40, 90
*        THE HEAD OF THE PRIME PHRASE FOUND
         BRANCH I-J-2, X, 60, 70
```

```
*       REDUCE  +F +I+I= I
        POINTPS → POINTPS + 1
        POSTFIX(POINTPS) → CRIDENT
        BRANCH 90
*       REDUCE  +P +I+I= +P+I
60      POINTPS → POINTPS + 1
        POSTFIX(POINTPS) → PSTACK(I)
        BRANCH 80
70      BRANCH PSTACK(I)=RPARENT, 80, X
*       REDUCE BINARY OPERATION
        POINTPS → POINTPS + I
        POSTFIX(POINTPS) → PSTACK(I-1)
80      I → I+1
        PSTACK(I) → NONTERM
        BRANCH I=2, X, 30
        BRANCH CRTOKEN=SCOL, X, 30
        POINTPS → POINTPS + I
        POSTFIX(POINTPS) → SCOL
        ECPARSE → 0
        BRANCH 0
90      ECPARSE → 11
        BRANCH 0
91      ECPARSE → 1
        BRANCH 0
92      ECPARSE → 2
UNLOADC POSTFIX(1),SPTFIX+3001,LPSTFIX
GOTO    ECPARSE, NEWTOKEN, X
EXIT    I
*
                                                                        CLSCHWD
UNIT    CLSCHWD
CALC    NC(SPSCHWD+2*I-1) → *,            *,
        NC(SPSCHWD+2*I) → *,            *,
UNLOADC SEARCHW(I), SPSCHWD+2*I+2999, 2
                                                                        DUMMY1
UNIT    DUMMY1
------------------------------------------------------------------------------- LEXI
*
*  LEXICAL ANALYZER FOR SEARCH PRESCRIPTION
*    STATE0 --- EXPECTING A NEW TOKEN
*    STATE1 --- UPPER CASE SHIFT CODE RECEIVED AS 1ST CODE
*    STATE2 --- RECEIVING LETTERS IN A SEARCH WORD
*    STATE3 --- UPPER CASE CODE IN ST2
                                                                        LEXI
UNIT    LEXI
                                                                  LPINWRD
ENTRY   LPINWRD
BREAK
LOADC   SPRESCH(1), SPSPRES+3001, 120
CALC    CPSP → CPSP+1
GOTO    CPSP>10, X, GETNSYM
```

```
CALC     WNSPRES → WNSPRES + 1
GOTO     WNSPRES>LWNPRES, LONGSP, X
CALC     CWSPRES → SPRESCR(WNSPRES)
CALC     CPSP → 1
                                                                GETNSYM
ENTRY    GETNSYM  $$ GET NEXT LETTER OF SEARCH PRESCRPT
CALC     CWSPRES → CWSPRES $CLS$ 6
         TCURSYM → CWSPRES $MASK$ 077
GOTO     TCURSYM=045, X, TAND  $$ ↑.OR↑.↑/
GOTO     STATE-1, X, GTST2, BACKUP, GTST2
CALC     CRTOKEN → 1     $$ CURRENT TOKEN = ↑.OR↑.
GOTO     EXITO
                                                                TAND
ENTRY    TAND
GOTO     TCURSYM=047, X, TLPAR
GOTO     STATE-1, X, GTST2, BACKUP, GTST2
CALC     CRTOKEN → 2     $$ CURRENT TOKEN = ↑.AND↑.
GOTO     EXITO
                                                                TLPAR
ENTRY    TLPAR
GOTO     TCURSYM=051, X, TRPAR
GOTO     STATE-1, X, GTST2, BACKUP, GTST2
CALC     CRTOKEN → 5     $$ CURRENT TOKEN = ↑.(↑.
GOTO     EXITO
                                                                TRPAR
ENTRY    TRPAR
GOTO     TCURSYM=052, X, TSCOLN         $$ ↑.)↑.↑/
GOTO     STATE-1, X, GTST2, BACKUP, GTST2
CALC     CRTOKEN → 6     $$ CURRENT TOKEN = ↑.)↑.
GOTO     EXITO
                                                                TSCOLN
ENTRY    TSCOLN
GOTO     TCURSYM=077, X, TBLANK
GOTO     STATE-1, X, GTST2, BACKUP, GTST2
CALC     CRTOKEN → 4     $$ CURRENT TOKEN = ↑.;↑.
GOTO     EXITO
                                                                TBLANK
ENTRY    TBLANK
GOTO     TCURSYM=055, X, TUCASE
GOTO     STATE-1, LPINWRD, GTST2, STORESW, GTST2
                                                                TUCASE
ENTRY    TUCASE
GOTO     TCURSYM=070, X, TNEGSYM  $$ UPPER CASE↑/
GOTO     STATE=3, GTST2, X
CALC     STATE → STATE + 1
GOTO     STORESW
                                                                TNEGSYM
ENTRY    TNEGSYM
GOTO     TCURSYM=042, X, NRESERV  $$ ↑.↑7↑.↑/
GOTO     STATE-1, GTST2, X, STORESW, NGST3
CALC     CRTOKEN → 3     $$ CURRENT TOKEN = ↑.↑7↑.
         CPSW → 1   $$ BACK UP CHAR POS OF SCH WRD
         STATE → 0
GOTO     EXITO
                                                                NGST3
```

```
ENTRY    NGST1
MOVE     BLANK, 10, SEARCHW(WPSW), CPSW-1
CALC     STATE ⇒ 1
GOTO     BACKUP1
                                                                    BACKUP
ENTRY    BACKUP
CALC     STATE ⇒ 0
                                                                    BACKUP1
ENTRY    BACKUP1
CALC     CRTOKEN ⇒ -WPSW      $$ CURRENT TOKEN = SEARCH WORD
         WPSW ⇒ WPSW + 1
         CPSP ⇒ CPSP-1
         CWSPRES ⇒ CWSPRES $CLS$ 54
         CPSW ⇒ 1
GOTO     WPSW>LWPSW, X, EXITO
WRITE    *ERROR*. *TOO MANY KEYWORDS. *PRESS -*N*E*X*T-.
GOTO     EREXIT
                                                                    NRESERV
ENTRY    NRESERV
GOTO     STATE-1, X, GTST2, STORESW, GTST2
                                                                    GTST2
ENTRY    GTST2
CALC     STATE ⇒ 2
                                                                    STORESW
ENTRY    STORESW
MOVE     TCURSYM,10,SEARCHW(WPSW),CPSW
CALC     CPSW ⇒ CPSW + 1
UNLOADC  SEARCHW(WPSW), SPSCHWD+2*WPSW+2999, 2
GOTO     CPSW>LSW, X, LPINWRD
WRITE    *ERROR*, *TOO LONG KEYWORD, *PRESS -*N*E*X*T-.
GOTO     EREXIT
                                                                    LONGSP
ENTRY    LONGSP
WRITE    *ERROR*. *TOO LONG SEARCH PRESCRIPTION. *PRESS -*N*E*X*T-.
                                                                    EREXIT
ENTRY    EREXIT
CALC     FCLEXI ⇒ -1
EXIT
                                                                    EXITO
ENTRY    EXITO
CALC     ECLEXI ⇒ 0
UNLOADC  SEARCHW(1),SPSCHWD+3001,40
EXIT     1

-------------------------------------------------------------------------------------------- EDITORS
*
                                                                          EDTLSP
UNIT     EDTLSP
CALC     J ⇒ 0
                                                                    EPINIT
ENTRY    EPINIT
LOADC    POSTFIX(1), SPTFIX+3001, LTSRNGE+LPSTFIX
CALC     TEMP ⇒ TSRANGE(-PSTACK(1)-LWPSW)
         K ⇒ 405
```

```
GOTO     TEMP=0,NOLESSN,X
AT       303
WRITE    ↑L↑E↑S↑S↑O↑N ↑N↑A↑M↑E
AT       325
WRITE    ↑A↑B↑S↑T↑R↑A↑C↑T
BREAK
                                                                                     NEXTBIT
ENTRY    NEXTBIT
CALC     J ⇐ J+1
GOTO     J>NICATLG, XEDLSP, X
CALC     TEMP ⇐ TEMP $CLS$ 1
         TEMP2 ⇐ TEMP $MASK$ 01
GOTO     TEMP2=1,X,NEXTBIT
BREAK
CALC     K ⇐ K+100
GOTO     K>3000, X, SHWDATA
PAUSE
ERASE
GOTO     FPINIT
                                                                                     SHWDATA
ENTRY    SHWDATA
AT       K
LOADC    LESSNM1(1).SUC1,LC1
SHOWA    LESSNM1(J)
AT       K+12
SHOWA    ABSTRCT(J),LABSTRC
GOTO     NEXTBIT
                                                                                     NOLESSN
ENTRY    NOLESSN
AT       1005
WRITE    ↑SORRY, BUT NO LESSONS SATISFY YOUR SPECIFICATION.  ↑M
         AKE LOOSE THE SPECIFICATION AND TRY AGAIN.
                                                                                     XEDLSP
ENTRY    XEDLSP
EXIT     1
*
*
*
                                                                                             EDTDCO
UNIT     EDTDCO
GOTO     I=0,ENOCORS,X
AT       304
WRITE    ↑L↑E↑S↑S↑O↑N          ↑T↑Y↑P↑E   ↑P↑E↑R↑F↑O↑R↑M ↑E↑X↑P↑C↑T↑D
AT       341
WRITE    ↑T↑I↑M↑E ↑R↑E↑Q↑R↑D       ↑D↑A↑T↑E
LOADC    COURCEN(1),SUCD,LCDIRCT
CALC     SDA ⇐ SCOUTLN+(PCOUTLN(I)-1)*WPLC
         NLSPEC ⇐ LCOUTLN(I)
         NWLOAD ⇐ LCOUTLN(I)*WPLC
JOIN     EDTDATA
EXIT     1
                                                                                     ENOCORS
ENTRY    ENOCORS
AT       1005
```

```
     LESSON GUIDED          AT      08.59.28.     ON      06/26/73

WRITE    +THE COURSE IS NOT IMPLEMENTED IN OUR DATA BASE.
EXIT     1
*
                                                                                    EDTDSR
UNIT     EDTDSR
AT       304
WRITE    +L+E+S+S+O+N          +T+Y+P+E     +P+E+R+F+O+R+M+A+N+C+E
AT       341
WRITE    +T+I+M+E +S+P+E+N+T      +D+A+T+E
LOADC    SOCSECN(1),SUB,NWLOAD
CALC     SDB ← SSRECRD+(PSRECRD(I)-1)*WPLC
         NWLOAD ← NLSREC*WPLC
JOIN     EDTDATA
EXIT     1
*
                                                                                    EDTDATA
UNIT     EDTDATA
CALC     J ← 0
         K ← 402
AT       3005
                                                                             NEXTLSN
ENTRY    NEXTLSN
BREAK
*
LOADC    NC(LC1+1)+SDB+NWLOAD
LOADC    LESSNM1(1),SOC1,LC1
CALC     J ← J+1
         K ← K+100
GOTO     J>NLSREC, RETURN, X
AT       K
SHOWA    LESSNM1(CLESSNN(J))
AT       K+46
SHOW     MONTH(J)
AT       K+49
SHOW     DAY(J)
AT       K+52
SHOW     YEAR(J)
AT       K+56
WRITE    /  /
AT       K+35
SHOW     TIME(J)
WRITE     MIN.
AT       K+23
SHOW     PERFORM(J)
GOTO     NEXTLSN
                                                                             RETURN
ENTRY    RETURN
EXIT     1

-------------------------------------------------------------------------------------- EDTDLD
*
*
*
```

```
                                                                                          EDTDLD
UNIT     EDTDLD
GOTO     I<0.FNDLESN.X
LOADC    LESSNM1(I).SDC1.LC1+LCATLG2+LKEYWDT
LOADC    LTYPE(0). SDLT, LLTYPE
CALC     TEMP ⇒ GLESSNN(I)
AT       504
WRITE    ↑L↑E↑S↑S↑O↑N ↑N↑A↑M↑E↑!
SHOWA    LESSNM2(I)
AT       530
WRITE    ↑T↑Y↑P↑E↑!
SHOWA    LTYPE(GTYPE(I))
AT       704
WRITE    ↑A↑B↑S↑T↑R↑A↑C↑T↑!
SHOWA    ABSTRCT(TEMP).LABSTRC
AT       1004
WRITE    ↑C↑A↑T↑E↑G↑O↑R↑Y↑!
CALC     TEMP2 ⇒ SUBJCAT(TEMP) $CLS$ 42
         J ⇒ 0
                                                                                   NEXTSJC
ENTRY    NEXTSJC
CALC     J ⇒ J+1
GOTO     J>NSUBJCT. DKEYWRD.X
CALC     TEMP2 ⇒ TEMP2 $CLS$ NDSUBJC*6
         TEMP3 ⇒ TEMP2 $MASK$ MASK2
GOTO     TEMP3=0. DKEYWRD.X
CALC     TEMP3 ⇒ TEMP3 $CLS$ 54
SHOWA    TEMP3.1
WRITE    .
SHOWA    TEMP2.NDSUBJC-1
WRITE    .
GOTO     NEXTSJC
                                                                                   DKEYWRD
ENTRY    DKEYWRD
AT       1204
WRITE    ↑K↑E↑Y↑W↑O↑R↑D↑S↑!
AT       1217
CALC     TEMP2 ⇒ KEYCODE(TEMP)
         J ⇒ 0
                                                                                   NEXTKEY
ENTRY    NEXTKEY
CALC     TEMP2 ⇒ TEMP2 $CLS$ NBKEYCD
         TEMP3 ⇒ TEMP2 $MASK$ MASK6
GOTO     TEMP3=0. DRQTIME. X
SHOWA    KEYWORD(TEMP3). LKEYWRD
CALC     J ⇒ J+1
GOTO     J>NKEYWRD. DRQTIME. X
WRITE
*  4 BLANKS
GOTO     NEXTKEY
                                                                                   DRQTIME
ENTRY    DRQTIME
AT       1604
WRITE    ↑R↑E↑Q↑U↑I↑R↑E↑D ↑T↑I↑M↑E↑!
```

```
        LESSON GUIDED        AT      08.59.28.    ON     06/24/73

SHOW     GTIME(I)
WRITE     MIN.
CALC     TEMP2 ⇒ RELATN(I)
         J ⇒ 0
         K ⇒ 1804
LOADC    CRELATN(0), SCRELCM, LREL
                                                                   XTREL
ENTRY    XTREL
CALC     TEMP2 ⇒ TEMP2 $CLS$ NBREL
         TEMP3 ⇒ TEMP2 $MASK$ 017
GOTO     TEMP3=0, XTDLD, X
AT       K
SHOWA    CRELATN(TEMP3)
WRITE    +1
*  WRITTEN TWO CHARS INCLUDING A BLANK AFTER +1
CALC     TEMP2 ⇒ TEMP2 $CLS$NBLN
         TEMP3 ⇒ TEMP2 $MASK$ 077
         K ⇒ K+12
AT       K
SHOWA    LESSNM1(TEMP3)
CALC     K ⇒ K+12
AT       K
SHOWA    ABSTRCT(TEMP3), 62
CALC     K ⇒ K+200
         K ⇒ K-24
GOTO     XTREL
                                                                   XTDLD
ENTRY    XTDLD
PAUSE
EXIT     1
                                                                   ENOLESN
ENTRY    ENOLESN
AT       1505
WRITE    +THE LESSON SPECIFIED IS NOT IMPLEMENTED IN OUR DATA BASE.
PAUSE
EXIT     1

--------------------------------------------------------------------------------------- INITIALIZE
*
                                                                         INPRECD
UNIT     INPRECD
CALC     PRECDNC(1) ⇒ 1
         PRECDNC(2) ⇒ -1
         PRECDNC(3) ⇒ -1
         PRECDNC(4) ⇒ 1
         PRECDNC(5) ⇒ -1
         PRECDNC(6) ⇒ 1
         PRECDNC(7) ⇒ -1
         PRECDNC(8) ⇒ 1
         PRECDNC(9) ⇒ 1
         PRECDNC(10) ⇒ -1
         PRECDNC(11) ⇒ 1
         PRECDNC(12) ⇒ -1
         PRECDNC(13) ⇒ 1
```

```
                 PRECONC(14) ⇐ -1
                 PRECONC(15) ⇐ 1
                 PRECONC(16) ⇐ 1
                 PRECONC(17) ⇐ 1
                 PRECONC(18) ⇐ 1
                 PRECONC(19) ⇐ 2
                 PRECONC(20) ⇐ 1
                 PRECONC(21) ⇐ 1
                 PRECONC(22) ⇐ -1
                 PRECONC(23) ⇐ -1
                 PRECONC(24) ⇐ -1
                 PRECONC(25) ⇐ 0
                 PRECONC(26) ⇐ -1
                 PRECONC(27) ⇐ 4
                 PRECONC(28) ⇐ -1
                 PRECONC(29) ⇐ -1
                 PRECONC(30) ⇐ -1
                 PRECONC(31) ⇐ -1
                 PRECONC(32) ⇐ 5
                 PRECONC(33) ⇐ -1
                 PRECONC(34) ⇐ 0
                 PRECONC(35) ⇐ -1
                 PRECONC(36) ⇐ 1
                 PRECONC(37) ⇐ 1
                 PRECONC(38) ⇐ 1
                 PRECONC(39) ⇐ 1
                 PRECONC(40) ⇐ 6
                 PRECONC(41) ⇐ 1
                 PRECONC(42) ⇐ 7
                 PRECONC(43) ⇐ 1
                 PRECONC(44) ⇐ 1
                 PRECONC(45) ⇐ 1
                 PRECONC(46) ⇐ 1
                 PRECONC(47) ⇐ 8
                 PRECONC(48) ⇐ 1
                 PRECONC(49) ⇐ 9
        UNLOADC  PRECONC(1),SPPRECD+3001,49
        EXIT     1
        *
        *
        *  INITIALIZE LESSON TYPE DECODER
                                                                      ILTDEC
        UNIT     ILTDEC
        CALC     LTYPE(0) ⇐ ↑,GENERAL↑,
                 LTYPE(1) ⇐ ↑,EXERCISE↑,
                 LTYPE(2) ⇐ ↑,EXAM↑,
        UNLOADC  LTYPE(0), SOLT, LLTYPE
        EXIT     1
        *
        *  ROUTINE FOR DEBUG
                                                                      DEBUG1
        UNIT     DEBUG1
        CALC     LESSNM2(1) ⇐ ↑,MONTECARLO↑,
                 LESSNM2(2) = +.PL1ARRAY  +.
                 LESSNM2(3) = +.PL1DATA   +.
                 LESSNM2(4) = +.PL1DO     +.
                 LESSNM2(5) = +.PL1IF     +.
                 LESSNM2(6) = +.PL1IO     +.
                 LESSNM2(7) = +.PL1LAB    +.
                 LESSNM2(8) = +.PL1X1     +.
                 LESSNM2(9) = +.RACETRACK +.
                 LESSNM2(10) = +.ROOTLAB   +.
                 LESSNM2(11) = +.SOMAGA    +.
                 NC(SCATLG2+2) = 00517000000000000000
                 NC(SCATLG2+4) = 01036000000000000000
                 NC(SCATLG2+6) = 00674000000000000000
                 NC(SCATLG2+8) = 00350000000000000000
                 NC(SCATLG2+10) = 00736041441400000000
                 NC(SCATLG2+12) = 00274000000000000000
                 NC(SCATLG2+14) = 01274000000000000000
                 NC(SCATLG2+16) = 01162000000000000000
                 NC(SCATLG2+18) = 00124000000000000000
                 NC(SCATLG2+20) = 01374000000000000000
                 NC(SCATLG2+22) = 00462000000000000000
        UNLOADC LESSNM2(1),SDC2,NICTLG2*NWLCLG2
        EXIT     1
        *
                                                                                          GRELCD
        UNIT     GRELCD
        CALC     CRELATN(0) = +.NO RELAT  +.
                 CRELATN(1) = +.PREREQSITE+.
                 CRELATN(2) = +.SEQUEL    +.
        UNLOADC CRELATN(0), SCRELCM, LREL
        EXIT     1

  ------------------------------------------------------------------------------------------ DEBUG1
        *
                                                                                          DEBUG3
        UNIT     DEBUG3
        CALC     NC(SKEYWDT+WPKKT*1-2) = +.ARRAY      +.
                 NC(SKEYWDT+WPKKT*1-1) = +.           +.
                 NC(SKEYWDT+WPKKT*1) = 00020000000000000000
                 NC(SKEYWDT+WPKKT*2-2) = +.ASSIGNMENT+.
                 NC(SKEYWDT+WPKKT*2-1) = +. STATEMENT+.
                 NC(SKEYWDT+WPKKT*2) = 0
                 NC(SKEYWDT+WPKKT*3-2) = +.DATA STRUC+.
                 NC(SKEYWDT+WPKKT*3-1) = +.TURE      +.
                 NC(SKEYWDT+WPKKT*3) = 00020000000000000000
                 NC(SKEYWDT+WPKKT*4-2) = +.DATA TYPE +.
                 NC(SKEYWDT+WPKKT*4-1) = +.          +.
                 NC(SKEYWDT+WPKKT*4) = 00100000000000000000
                 NC(SKEYWDT+WPKKT*5-2) = +.DECLARATIO+.
                 NC(SKEYWDT+WPKKT*5-1) = +.N         +.
                 NC(SKEYWDT+WPKKT*5) = 00100000000000000000
                 NC(SKEYWDT+WPKKT*6-2) = +.FORTRAN   +.
                 NC(SKEYWDT+WPKKT*6-1) = +.          +.
```

```
                NC(SKEYWDT+WPKKT*6) ⇒ 0
                NC(SKEYWDT+WPKKT*7-2) ⇒ ↑,GAME       ↑,
                NC(SKEYWDT+WPKKT*7-1) ⇒ ↑,           ↑,
                NC(SKEYWDT+WPKKT*7) ⇒ 04400000000000000000
                NC(SKEYWDT+WPKKT*8-2) ⇒ ↑,INPUT      ↑,
                NC(SKEYWDT+WPKKT*8-1) ⇒ ↑,           ↑,
                NC(SKEYWDT+WPKKT*8) ⇒ 02000000000000000000
                NC(SKEYWDT+WPKKT*9-2) ⇒ ↑,ITERATION ↑,
                NC(SKEYWDT+WPKKT*9-1) ⇒ ↑,           ↑,
                NC(SKEYWDT+WPKKT*9) ⇒ 01020000000000000000
                NC(SKEYWDT+WPKKT*10-2) ⇒ ↑,LABEL      ↑,
                NC(SKEYWDT+WPKKT*10-1) ⇒ ↑,           ↑,
                NC(SKEYWDT+WPKKT*10) ⇒ 0
                NC(SKEYWDT+WPKKT*11-2) ⇒ ↑,MANAGEMENT↑,
                NC(SKEYWDT+WPKKT*11-1) ⇒ ↑,           ↑,
                NC(SKEYWDT+WPKKT*11) ⇒ 00400000000000000000
                NC(SKEYWDT+WPKKT*12-2) ⇒ ↑,NUMERICAL ↑,
                NC(SKEYWDT+WPKKT*12-1) ⇒ ↑,METHOD     ↑,
                NC(SKEYWDT+WPKKT*12) ⇒ 00202000000000000000
                NC(SKEYWDT+WPKKT*13-2) ⇒ ↑,OUTPUT     ↑,
                NC(SKEYWDT+WPKKT*13-1) ⇒ ↑,           ↑,
                NC(SKEYWDT+WPKKT*13) ⇒ 02000000000000000000
                NC(SKEYWDT+WPKKT*14-2) ⇒ ↑,PL/1       ↑,
                NC(SKEYWDT+WPKKT*14-1) ⇒ ↑,           ↑,
                NC(SKEYWDT+WPKKT*14) ⇒ 03174000000000000000
                NC(SKEYWDT+WPKKT*15-2) ⇒ ↑,PROGRAMMIN↑,
                NC(SKEYWDT+WPKKT*15-1) ⇒ ↑,G LANGUAGE↑,
                NC(SKEYWDT+WPKKT*15) ⇒ 03574000000000000000
                NC(SKEYWDT+WPKKT*16-2) ⇒ ↑,PROGRAM FL↑,
                NC(SKEYWDT+WPKKT*16-1) ⇒ ↑,OW CONTROL↑,
                NC(SKEYWDT+WPKKT*16) ⇒ 01040000000000000000
                NC(SKEYWDT+WPKKT*17-2) ⇒ ↑,PRACTICE   ↑,
                NC(SKEYWDT+WPKKT*17-1) ⇒ ↑,           ↑,
                NC(SKEYWDT+WPKKT*17) ⇒ 00006000000000000000
                NC(SKEYWDT+WPKKT*18-2) ⇒ ↑,RECURTION ↑,
                NC(SKEYWDT+WPKKT*18-1) ⇒ ↑,           ↑,
                NC(SKEYWDT+WPKKT*18) ⇒ 0
                NC(SKEYWDT+WPKKT*19-2) ⇒ ↑,ROOT FINDI↑,
                NC(SKEYWDT+WPKKT*19-1) ⇒ ↑,NG         ↑,
                NC(SKEYWDT+WPKKT*19) ⇒ 00002000000000000000
                NC(SKEYWDT+WPKKT*20-2) ⇒ ↑,SIMULATION↑,
                NC(SKEYWDT+WPKKT*20-1) ⇒ ↑,           ↑,
                NC(SKEYWDT+WPKKT*20) ⇒ 04600000000000000000
      UNLOADC KEYWORD(1),SDKY,LKEYWDT
      EXIT    1

   ------------------------------------------------------------------------------------------ DEBUG2
      *

                                                                                 DBGCLG1
      UNIT     DBGCLG1
      CALC     LESSNM1(1) ⇒ ↑,RACETRACK ↑,
               LESSNM1(2) ⇒ ↑,PLITO      ↑,
               LESSNM1(3) ⇒ ↑,PLIDO      ↑,
               LESSNM1(4) ⇒ ↑,SOMAGA     ↑,
                LESSNM1(5) ⇐ *.MONTECARLO*.
                LESSNM1(6) ⇐ *.PL1DATA   *.
                LESSNM1(7) ⇐ *.PL1IF     *.
                LESSNM1(8) ⇐ *.PL1ARRAY  *.
                LESSNM1(9) ⇐ *.PL1X1     *.
                LESSNM1(10) ⇐ *.PL1LAB    *.
                LESSNM1(11) ⇐ *.ROOTLAB   *.
        PACK    TEMP.ABSTRCT(1).*SIMULATION *EXPERIMENT
        PACK    TEMP.ABSTRCT(2).*P*L/1 *INPUT/*OUTPUT
        PACK    TEMP.ABSTRCT(3).*INTRODUCTION TO *P*L/1 *I*O-STATEMENTS
        PACK    TEMP.ABSTRCT(4).*SOFTWARE *MANAGEMENT *GAME TO TEACH PROGRAMMING
        PACK    TEMP.ABSTRCT(5).*CALCULATION OF AREA BY *MONTE *CARLO METHOD
        PACK    TEMP.ABSTRCT(6).*BEGINNING COMPUTER SCIENCE LESSON
        PACK    TEMP.ABSTRCT(7).*P*L/1 *I*F-*T*H*E*N-*E*L*S*E STATEMENTS
        PACK    TEMP.ABSTRCT(8).*INTRODUCTION TO *P*L/1 ARRAYS
        PACK    TEMP.ABSTRCT(9).*THE EXAM COVERING THE MATERIAL IN PL1 SEQUENCE
        PACK    TEMP.ABSTRCT(10).A LAB ALLOWING STUDENT TO WRITE SIMPLE PL1 PROG
        PACK    TEMP.ABSTRCT(11).*LESSON TO WORK WITH FOUR ROOT FINDING METHODS
        CALC    KEYCODE(1) ⇐ 00500340000000000000
                KEYCODE(2) ⇐ 00340401500000000000
                KEYCODE(3) ⇐ 00340441700000000000
                KEYCODE(4) ⇐ 00500542001600000000
                KEYCODE(5) ⇐ 00500600000000000000
                KEYCODE(6) ⇐ 00340200504000000000
                KEYCODE(7) ⇐ 00340740000000000000
                KEYCODE(8) ⇐ 00340140102200000000
                KEYCODE(9) ⇐ 00341000000000000000
                KEYCODE(10) ⇐ 00341040000000000000
                KEYCODE(11) ⇐ 00301142100000000000
        UNLOADC LESSNM1(1).SDC1.LC1
        EXIT    1
        *
        *  INITIALIZE COURSE DIRECTORY FOR DEBUG

                                                                        DEBUG6
        UNIT    DEBUG6
        CALC    COURCEN(1) ⇐ *.CS101E1   *.
                COURCEN(2) ⇐ *.CS101N1   *.
                COURCEN(3) ⇐ *.CS102B1   *.
                COURCEN(4) ⇐ *.CS103M1   *.
                COURCEN(5) ⇐ *.CS104K1   *.
                COURCEN(6) ⇐ *.CS105A1   *.
                COURCEN(7) ⇐ *.CS105O1   *.
                COURCEN(8) ⇐ *.CS106F1   *.
                COURCEN(9) ⇐ *.CS107P1   *.
                COURCEN(10) ⇐ *.CS108H1   *.
                NC(3*1+SCDIRCT) ⇐ 00014131300000100 5005
                NC(3*2+SCDIRCT) ⇐ 00000000000000601 0010
                NC(3*3+SCDIRCT) ⇐ 00000000000001600 3003
                NC(3*4+SCDIRCT) ⇐ 00000000000002100 4004
                NC(3*5+SCDIRCT) ⇐ 00000000000002500 6006
                NC(3*6+SCDIRCT) ⇐ 00001131300003300 5005
                NC(3*7+SCDIRCT) ⇐ 00000000000004000 4004
                NC(3*8+SCDIRCT) ⇐ 00000000000004400 3003
                NC(3*9+SCDIRCT) ⇐ 00000000000004700 5005
                NC(3*10+SCDIRCT) ⇒ 00000000000054003003
     UNLOADC  COURCEN(1),SDCD,LCDIRCT
     EXIT     1
     *
                                                                                   DEBUGS
     UNIT     DEBUGS
     BREAK
     LOADC    SEARCHW(1),SPSCHWD+3001,109
     SHOW     POSTFIX(I)
                                                                                   DEBUG2
     UNIT     DEBUG2
     LOADC    SEARCHW(1),SPSCHWD+3001,40
     SHOWA    SEARCHW(I)
     SHOWA    NC(2*I+SPSCHWD)
                                                                                   DUMMY
     UNIT     DUMMY
 ------------------------------------------------------------------------------------------ DEBUG3
     *
                                                                                   DEBUG7
     UNIT     DEBUG7
     CALC     SOCSFCN(1) ⇒ ↑,031230729 ↑,
              SOCSFCN(2) ⇒ ↑,216338201 ↑,
              SOCSFCN(3) ⇒ ↑,216338834 ↑,
              SOCSFCN(4) ⇒ ↑,346520078 ↑,
              SOCSFCN(5) ⇒ ↑,573443792 ↑,
              SOCSFCN(6) ⇒ ↑,100250957 ↑,
              SOCSFCN(7) ⇒ ↑,101428463 ↑,
              SOCSFCN(8) ⇒ ↑,216338201 ↑,
              SOCSFCN(9) ⇒ ↑,352074591 ↑,
              SOCSFCN(10) ⇒ ↑,352075612 ↑,
              SOCSFCN(11) ⇒ ↑,471930062 ↑,
              SOCSFCN(12) ⇒ ↑,763496623 ↑,
              SOCSFCN(13) ⇒ ↑,882438257 ↑,
              SOCSFCN(14) ⇒ ↑,014057223 ↑,
              SOCSFCN(15) ⇒ ↑,655328344 ↑,
              SOCSFCN(16) ⇒ ↑,674912645 ↑,
     PACK     TEMP,SNAME(1),RESTON JAMES
     PACK     TEMP,SNAME(2),BERGMAN INGRID
     PACK     TEMP,SNAME(3),EINSTEIN ALBERT
     PACK     TEMP,SNAME(4),FONDA JANE
     PACK     TEMP,SNAME(5),BOULEZ PIERRE
     CALC     NC(WPS*4) ⇒ NC(WPS*4) $MASK$ MASKE $UNION$ 01
              NC(WPS*5) ⇒ NC(WPS*5) $MASK$ MASKE $UNION$ 013
     UNLOADC  SOCSFCN(1),SSDIRCT,NSDIRCT
     EXIT     1
     *
                                                                                   DBGSCO
     UNIT     DBGSCO
     CALC     NC(SAD+1) ⇒ 00436000000022711124
```

```
          NC(SA0+2) ⇐ 00550000000000111134
          NC(SA0+3) ⇐ 00443000000021511160
          NC(SA0+4) ⇐ 00750000000000111172
          NC(SA0+5) ⇐ 01074000000000111212
          NC(SA0+6) ⇐ 01124000000000111224
          NC(SA0+7) ⇐ 01231000000000111244
          NC(SA0+8) ⇐ 01317000000000111255
          NC(SA0+9) ⇐ 00150000000017111270
          NC(SA0+10) ⇐ 00250000000020111301
          NC(SA0+11) ⇐ 00343000000024111312
UNLOADC NC(SA0+1),SCOUTLN,11
EXIT      1
*

                                                                      DBGSSR
UNIT      DBGSSR
CALC      NC(1) ⇐ 00136000000022711476
          NC(2) ⇐ 00243000000021511505
          NC(3) ⇐ 00350000000030111517
          NC(4) ⇐ 00424000000016311524
          NC(5) ⇐ 00555000000000111545
          NC(6) ⇐ 00643000000024111552
          NC(7) ⇐ 00750000000000111562
          NC(8) ⇐ 01000000000000000000
          NC(9) ⇐ 01100000000000000000
          NC(10) ⇐ 01200000000000000000
          NC(11) ⇐ 01300000000000000000
UNLOADC NC(1),SSRECRD,11
*  DEBUG ROUTINE

                                                                      DEBUG5
UNIT      DEBUG5
AT        3005
WRITE     SEARCH RANGE VECTOR =
LOADC     POSTFIX(1), SPTFIX+3001, LTSRNGE+LPSTFIX
SHOWO     TSRANGE(-PSTACK(1)-LWPSW)
*  CODE FOR DEBUG

                                                                      DEBUG4
UNIT      DEBUG4
AT        1505
WRITE     POSTFIX ⇐
DO        DEBUG5, I⇐1,20
AT        1805
WRITE     SEARCH WORD TABLE ⇐
DO        DEBUG2, I⇐1,10
EXIT      1
```

```
  LESSON GUIDED        AT      08.59.28.     ON      06/24/73

UNIT      BLOCK       UNIT LOCATION        REFERENCES TO UNIT

AND       SLPRESC            502               498
BACKUP    LEXI               802               760        765        770        775        780
BACKUP1   LEXI               804               801
BUP       SLPRESC            457               452        452
BSEARCH   SEARCH             588               338        420        463        477
CALCSRV   SLPRESC            447               262
CLSCHWD   PARSER             734               672
CMPT      SEARCH             599               638
CSTTEM    SEARCH             590
DBGCLG1   DEBUG2            1183               199
DBGSCO    DEBUG3            1283               200
DBGSSR    DEBUG3            1298               201
DEBUGS    DEBUG2            1245              1321
DEBUG1    INITIALIZE        1084               206
DEBUG2    DEBUG2            1249              1324
DEBUG3    DEBUG1            1118               208
DEBUG4    DEBUG3            1318               256
DEBUG5    DEBUG3            1312               266
DEBUG6    DEBUG2            1221               207
DEBUG7    DEBUG3            1256               209
DECRJ     SLPRESC            504               501
DEFINE1   DEFINE1              9
DKEXIT    IDECODE            316               312
DKEYTBL   IDECODE            300               240        279        483
DKEYWRD   EDTDLD             967               957        960
DROTIME   EDTDLD             983               976        979
DUMMY     DEBUG2            1253
DUMMY1    PARSER             738
EDTDATA   EDITORS            903               885        900
EDTDCO    EDITORS            875               367
EDTDLD    EDTDLD             937               341
EDTDSR    EDITORS            892               433
EDTLSP    EDITORS            835               272
EILGINE   SLPRESC            535               452
ENDCORS   EDITORS            887               876
ENDFILE   SEARCH             640
ENDLESN   EDTDLD            1014               938
EPINIT    EDITORS            837               858
EPSTOVF   SLPRESC            522               454
ERFXIT    LEXI               826               812        823
ERNOCRS   CONTROL            438               385
ERR*/*/              NOT FOUND                 232
ETSROVF   SLPRESC            526               497
EXITO     LEXI               829               762        767        772        777        782        797        810
GETNSYM   LEXI               756               751
GRFLCD    INITIALIZE        1110               203
GTST2     LEXI               815               760        760        765        765        770        770        775
                                               775        780        780        785        785        788        793
                                               814        814

HSPRESC   IDECODE            278               239
IDCDCO    CONTROL            347               230
IDCDLD    CONTROL            321               230
IDCDSR    CONTROL            373               230
IDCLSP    IDECODE            238               230        490
IDECODE   IDECODE            215               211        275        343        370        436        442
```

```
   LESSON GUIDEO          AT      08.59.28.     ON       06/24/73

UNIT       BLOCK       UNIT LOCATION        REFERENCES TO UNIT

ILTDEC     INITIALIZE       1076                204
INCRP      SLPRESC           449                456         516          538
INCRW      SLPRESC           494                481
INDRECD    INITIALIZE       1021                196
IPLUSI     IDECODE           310                306
ISRNGE1    SLPRESC           469                458
ISRNGE2    SLPRESC           491                472
LEXI       LEXI              746                680
LONGSP     LEXI              824                753
LPINWRD    LEXI              747                785         821
LSRANGE    SLPRESC           539
NEG        SLPRESC           507                498
NEWTOKE               NOT FOUND                 731
NEWTOKEN   PARSER            679
NEXTBIT    EDITORS           847                852         865
NEXTITM    PARSER            655                661
NEXTKEY    EDTOLD            973                982
NEXTLSN    EDITORS           907                930
NEXTSJC    EDTOLD            955                966
NFOUND     PARSER            663                657
NGST3      LEXI              798                793
NOKEY      SLPRESC           482                464         478
NOLESSN    EDITORS           866                841
NRESERV    LEXI              813                792
NXTKEYW    IDECODE           304                315
OPRND2     SLPRESC           471                452         452          467
PARSER     PARSER            671                251
RETURN     EDITORS           931                914
SCOLN      SLPRESC           517                498
SETPARM    CALCSPV2          542                462         476
SHWDATA    EDITORS           859                855
SKEXIT     SLPRESC           529                518         521          525
SPCDIRC    CALCSPV2          556                353         380
SSEARCH    PARSER            649                355         382
STACKO     SLPRESC           509                506
STORESW    LEXI              817                785         790          793          814
TAND       LEXI              763                759
TBLANK     LEXI              783                779
TLPAR      LEXI              768                764
TNEGSYM    LEXI              791                787
TRPAR      LEXI              773                769
TSCOLN     LEXI              778                774
TUCASE     LEXI              786                784
XEOLSP     EDITORS           870                849
XTOLD      EDTOLD           1011                995
XTREL      EDTOLD            992               1010
```

```
  06/24/73  UNIV OF ILL CY73-24 VER 05.50 06/16/73

08.59.26.JOB
08.59.27.ATTACH(TPH)
08.59.27.ATTACH(TPRINTS)
08.59.27.LINK.
08.59.27.MAP.OFF.
08.59.27.REDUCE.
08.59.27.TPB.TPRINTS.OUTPUT.
08.59.35.CP  000.431 SEC.
08.59.35.PP  007.821 SEC.
```

```
FDPRI35  ///// END OF LIST /////
```

| | | | |
|---|---|---|---|
| **BIBLIOGRAPHIC DATA SHEET** | 1. Report No. UIUCDCS-R-73-587 | 2. | 3. Recipient's Accession No. |

| | |
|---|---|
| 4. Title and Subtitle<br>GUIDE-O<br>-- AN EXPERIMENTAL INFORMATION SYSTEM -- | 5. Report Date<br>August, 1973 |
| | 6. |
| 7. Author(s)<br>Shinnichi Murai | 8. Performing Organization Rept. No. |
| 9. Performing Organization Name and Address<br>Department of Computer Science<br>University of Illinois at Urbana-Champaign<br>Urbana, Illinois 61801 | 10. Project/Task/Work Unit No. |
| | 11. Contract/Grant No.<br>NSF GJ-31222 |
| 12. Sponsoring Organization Name and Address<br>National Science Foundation<br>Washington, D.C. | 13. Type of Report & Period Covered |
| | 14. |

15. Supplementary Notes

16. Abstracts

The experimental information system GUIDE-O is a bibliographic aid for those students who are taking the introductory computer science courses some of the material of which are implemented as PLATO-IV lessons. The functions, the data base and the detailed description of each module of the system are presented.

17. Key Words and Document Analysis. 17a. Descriptors

information retrieval, information systems
conversational information system
interactive information system
computer-assisted instruction
data base

17b. Identifiers/Open-Ended Terms

17c. COSATI Field/Group

| | | |
|---|---|---|
| 18. Availability Statement<br>release unlimited | 19. Security Class (This Report)<br>UNCLASSIFIED | 21. No. of Pages<br>101 |
| | 20. Security Class (This Page<br>UNCLASSIFIED | 22. Price |

10-5-73